EECE 5639 Computer Vision I

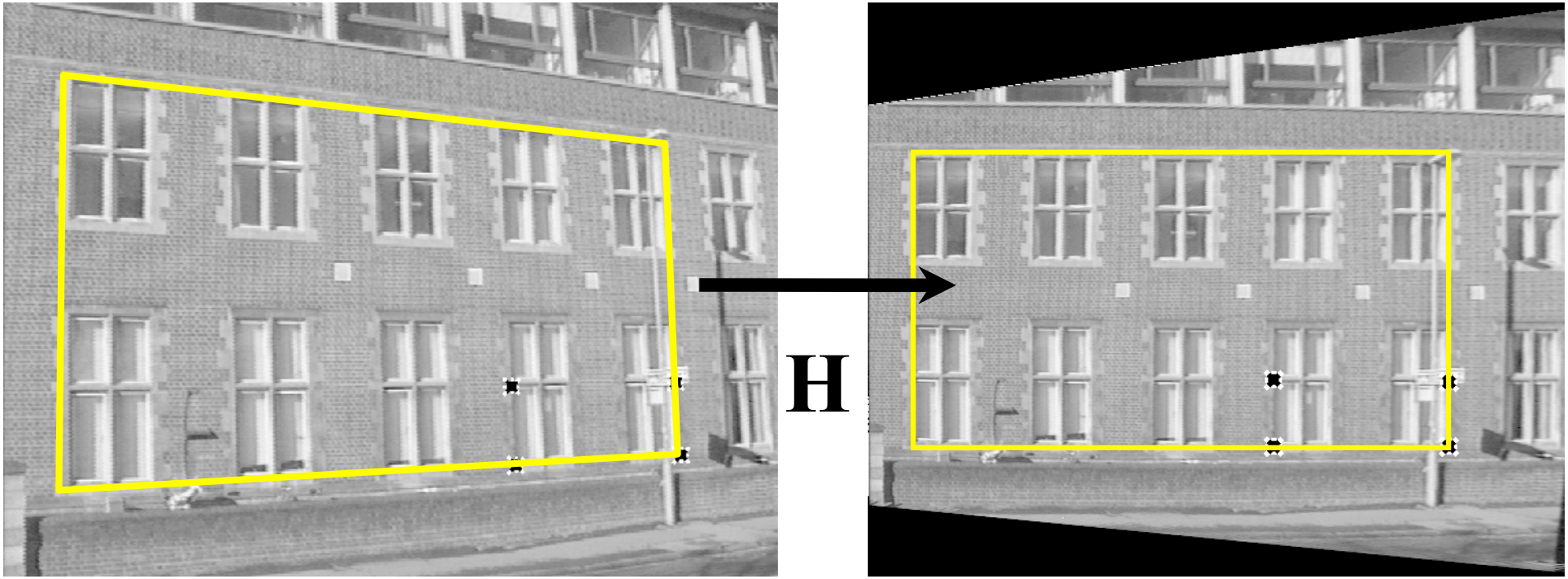
Lecture 14

- **Planar Unwarping**
- **Stereo**
- **Project 2 is out …**

Next Class

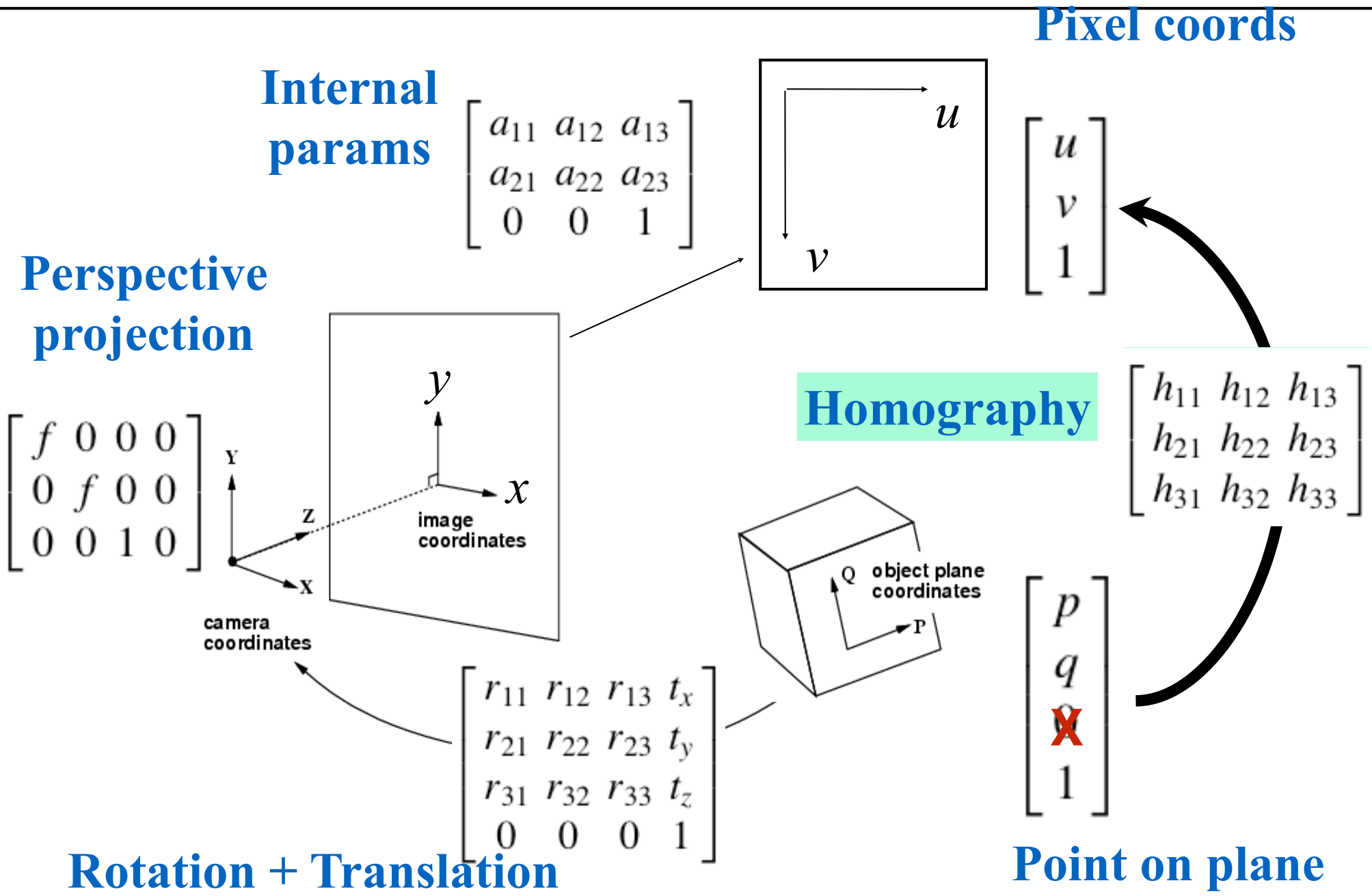
- **Stereo**

Projective Warping Example

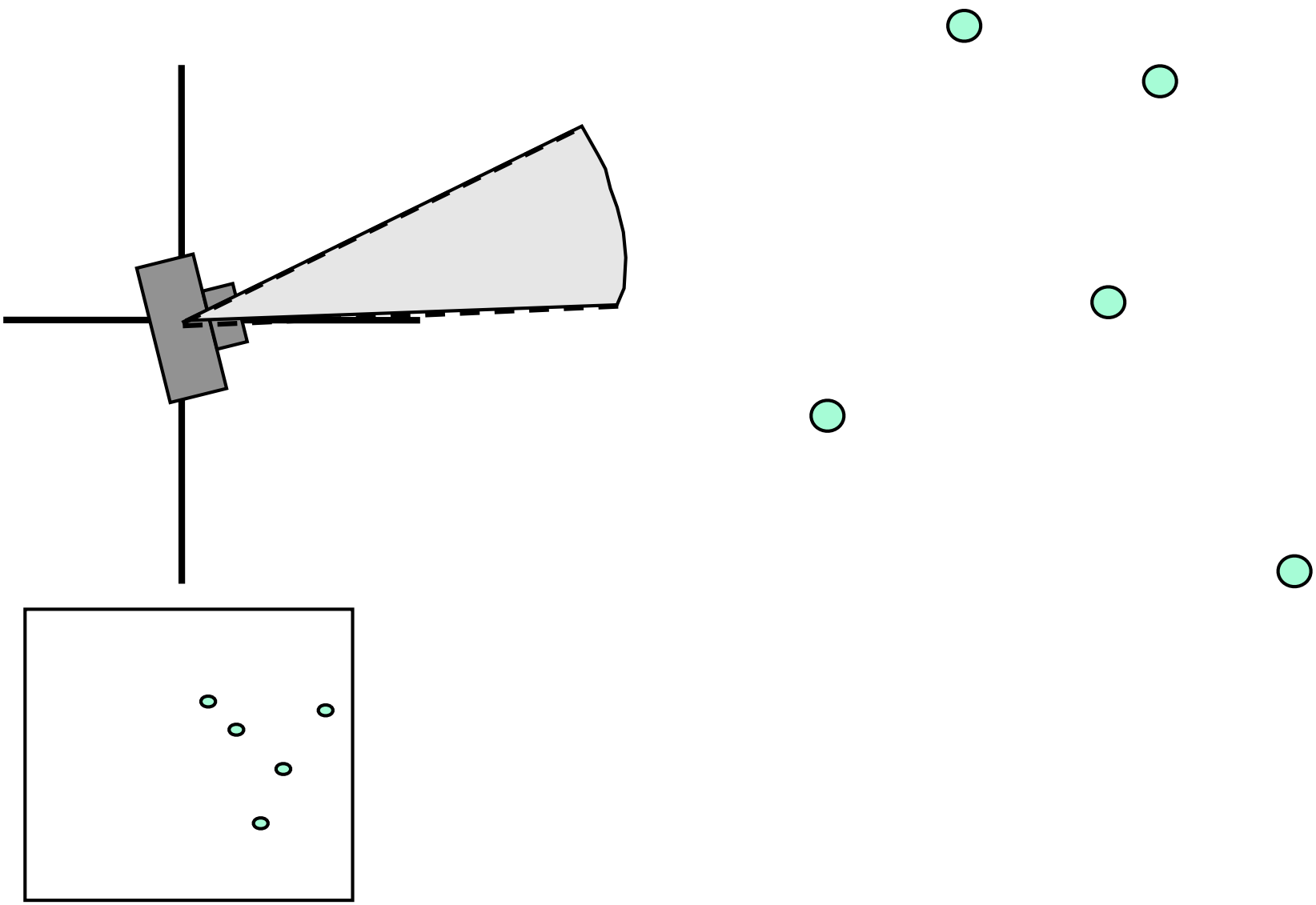


from Hartley & Zisserman

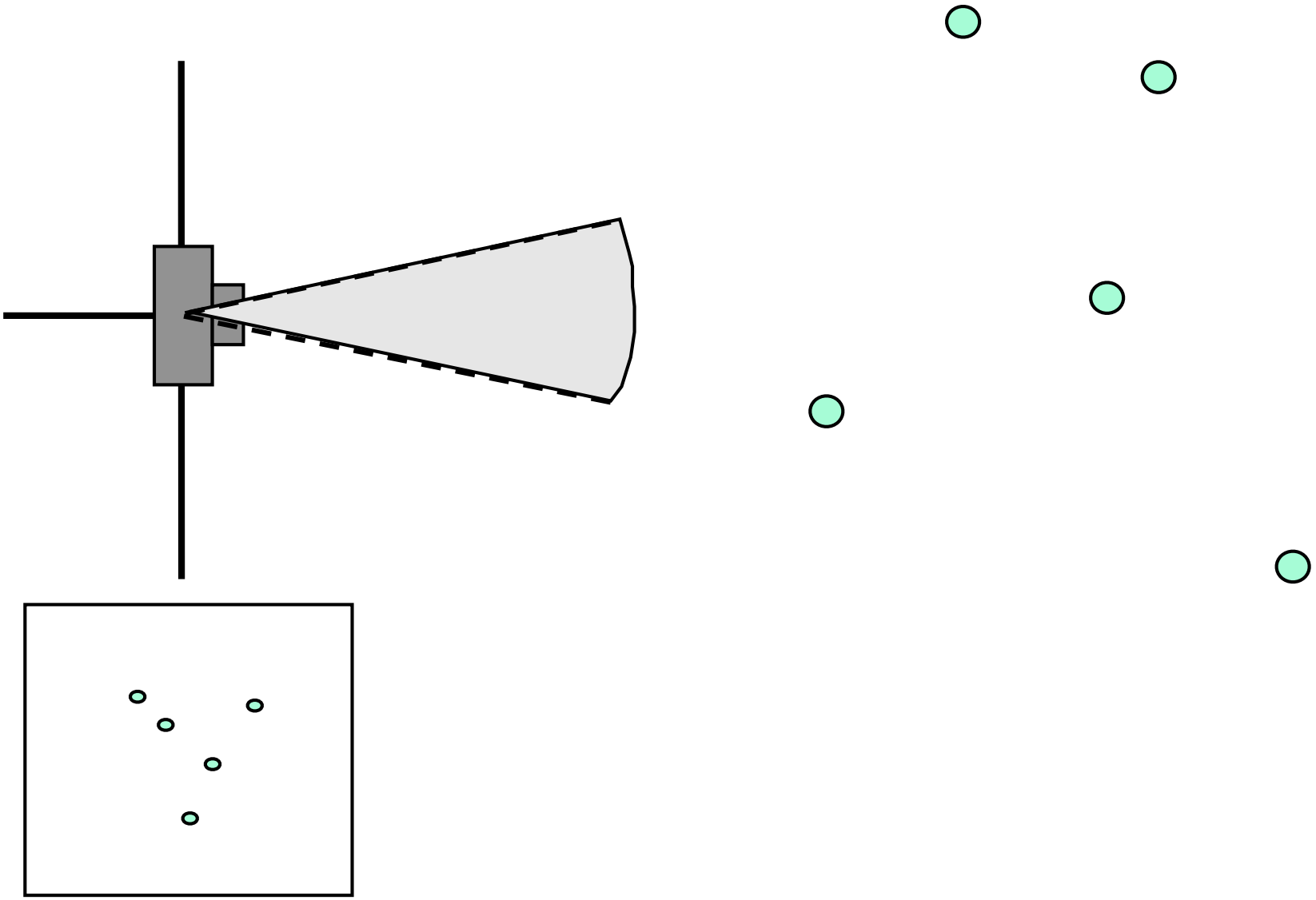
Summary: Planar Projection



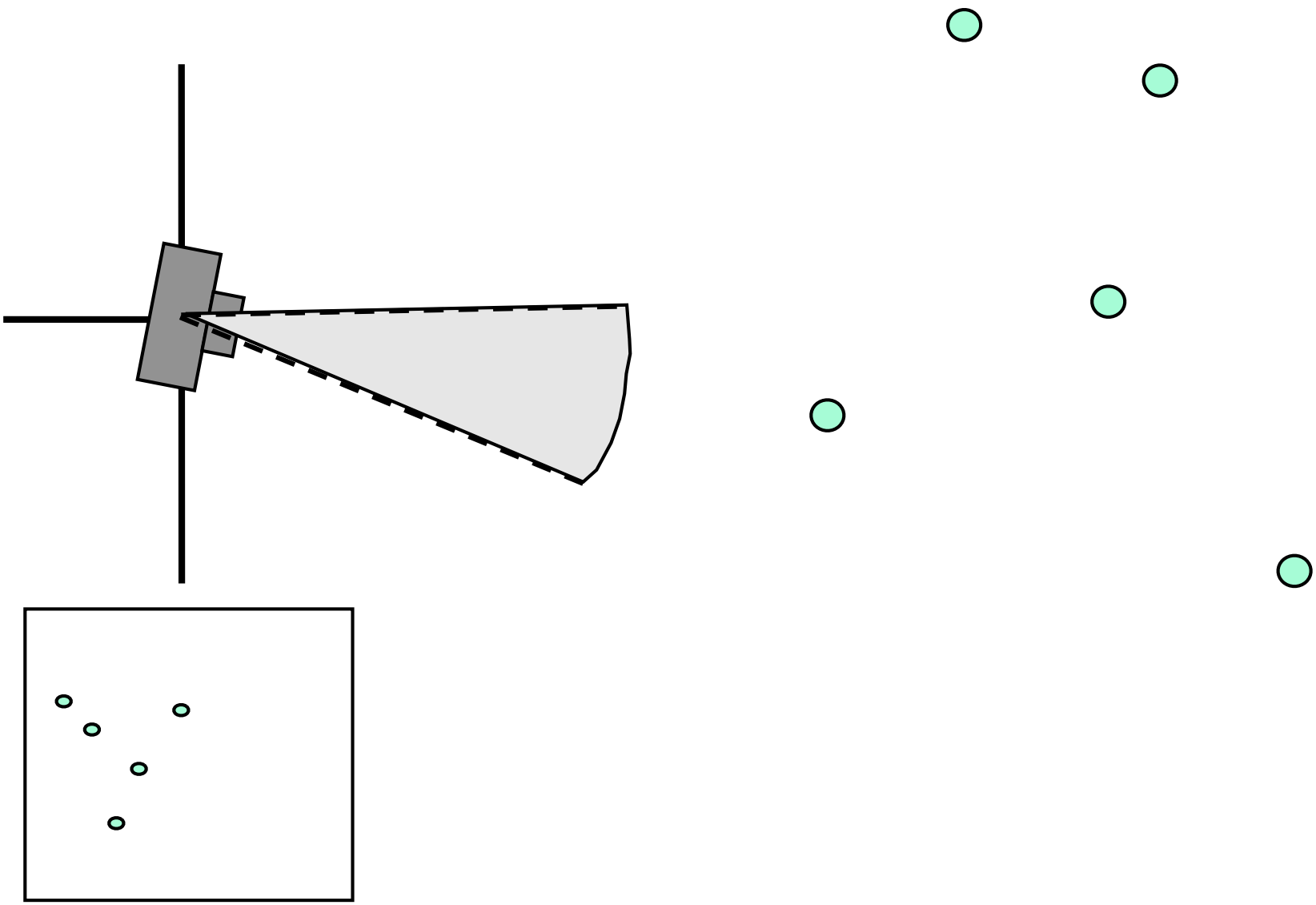
Rotating Camera (top-down view)



Rotating Camera (top-down view)

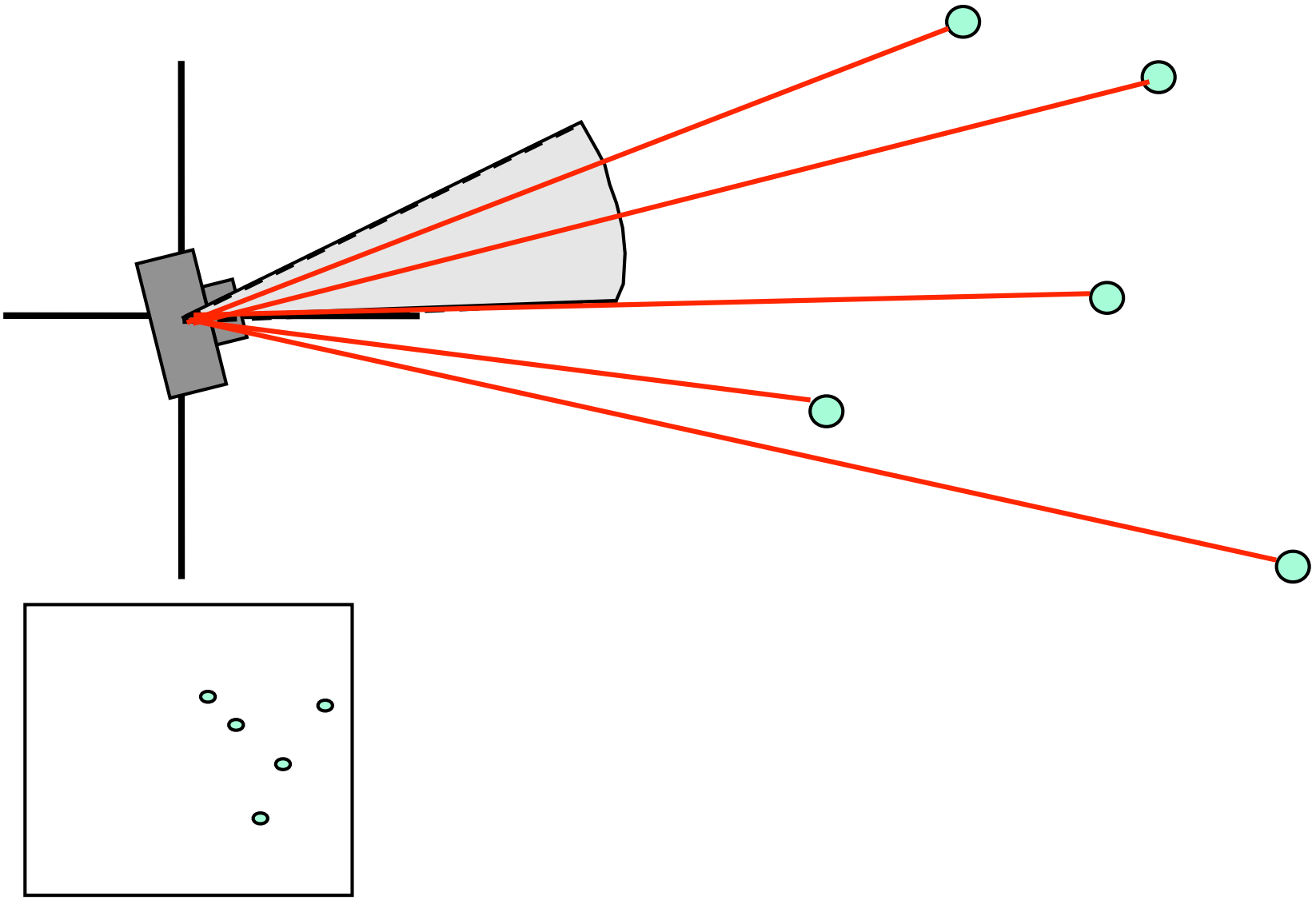


Rotating Camera (top-down view)



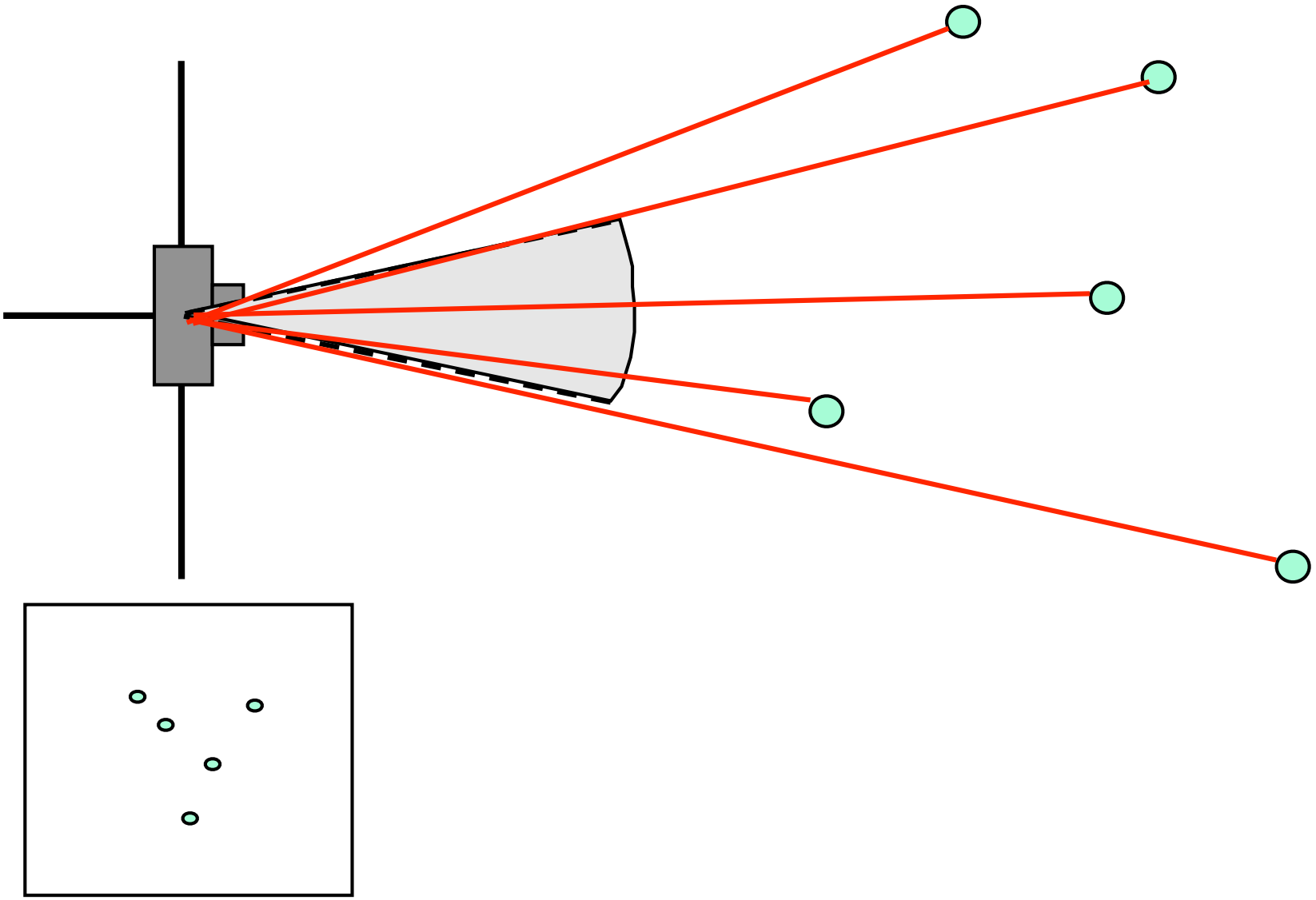
Rotating Camera (top-down view)

Rays in camera coord system are invariant!



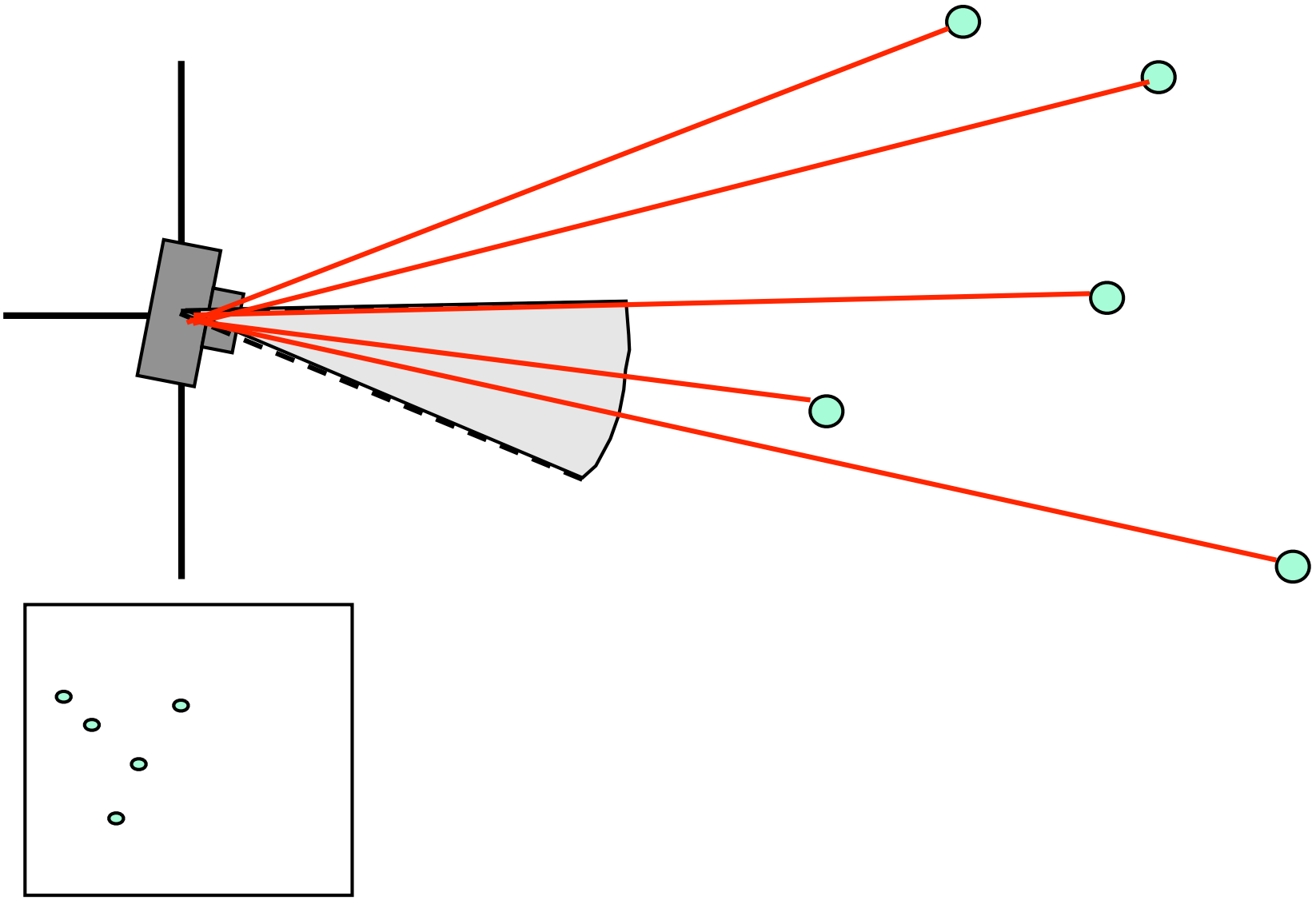
Rotating Camera (top-down view)

Rays in camera coord system are invariant!

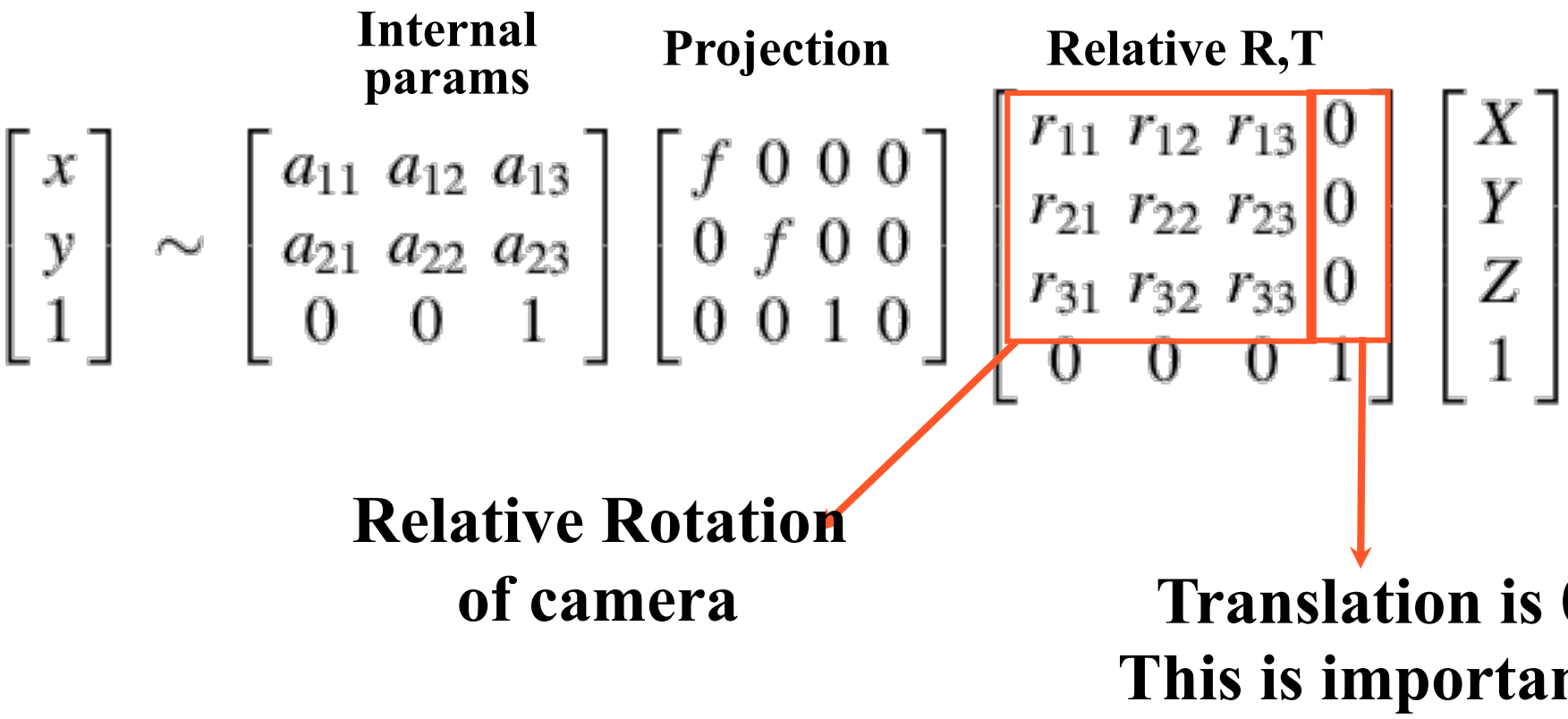


Rotating Camera (top-down view)

Rays in camera coord system are invariant!



Special Case : Rotating Camera



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Special Case : Rotating Camera

Internal params **Projection** **Relative R,T**

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Relations among Images Taken by Rotating Camera

Image 1

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Same ray!

Image 2

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h'_{11} & h'_{12} & h'_{13} \\ h'_{21} & h'_{22} & h'_{23} \\ h'_{31} & h'_{32} & h'_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} h'_{11} & h'_{12} & h'_{13} \\ h'_{21} & h'_{22} & h'_{23} \\ h'_{31} & h'_{32} & h'_{33} \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Practical Issues

How to:

- Find four or more point correspondences
- Estimate the homography given the point correspondences.
- (Un)warp image pixel values to produce a new picture.

Normalization

•When a scene is imaged by different sensors, or under different illumination intensities, both the SSD and the C_{fg} can be large for windows representing the same area in the scene!

•A solution is to **NORMALIZE** the pixels in the windows before comparing them (take away the mean and scale with the std. dev):

$$\hat{f} = \frac{f}{\|f\|} = \frac{f}{\sqrt{\sum_{[i,j]\in R} f^2(i,j)}}$$

$$\hat{g} = \frac{g}{\|g\|} = \frac{g}{\sqrt{\sum_{[i,j]\in R} g^2(i,j)}}$$

Normalized Cross-Correlation N_{fg}

$$N_{fg} = C_{\hat{f}\hat{g}} = \sum_{[i,j]\in R} \hat{f}(i,j)\hat{g}(i,j)$$

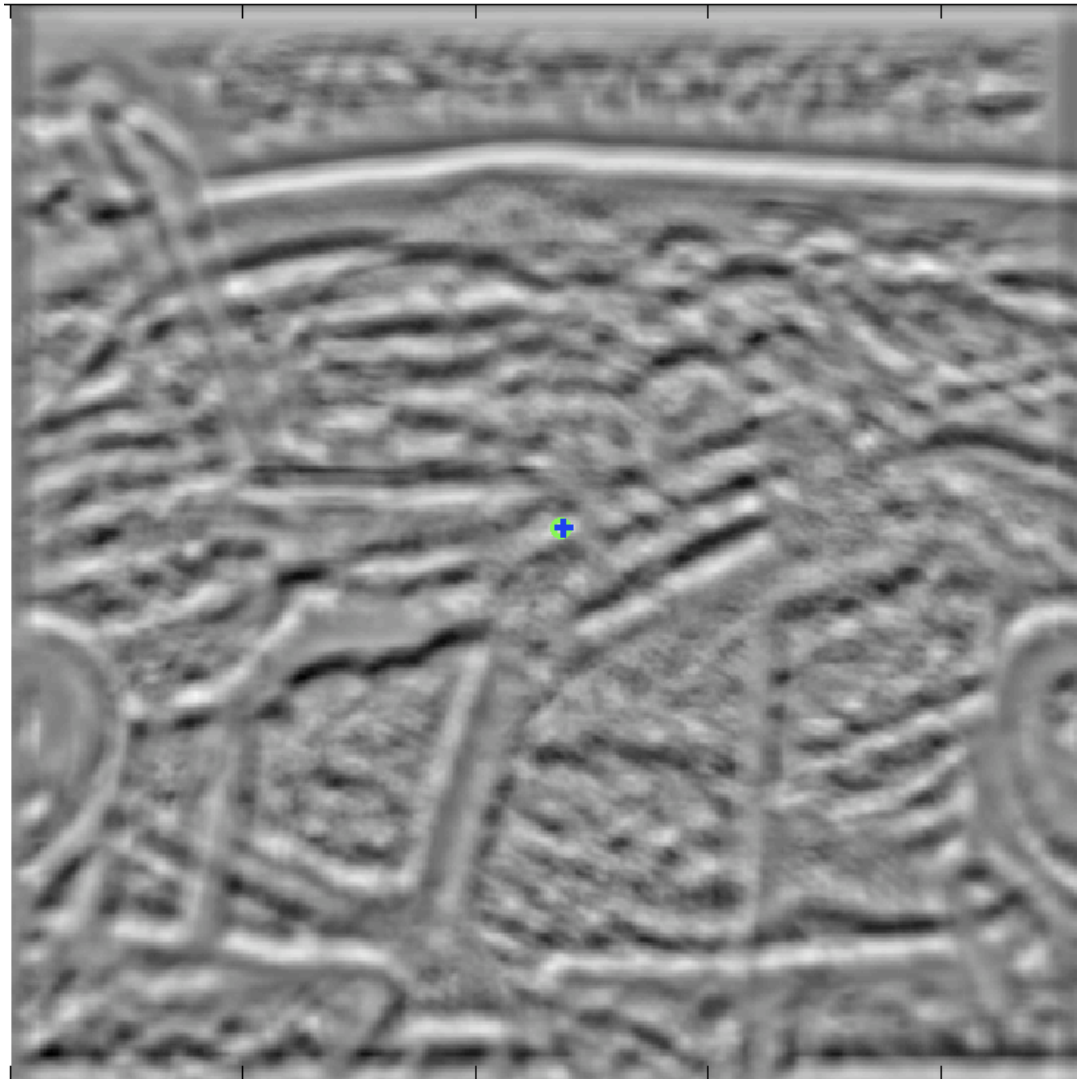
$$\hat{f} = \frac{f}{\|f\|} \quad \hat{g} = \frac{g}{\|g\|}$$

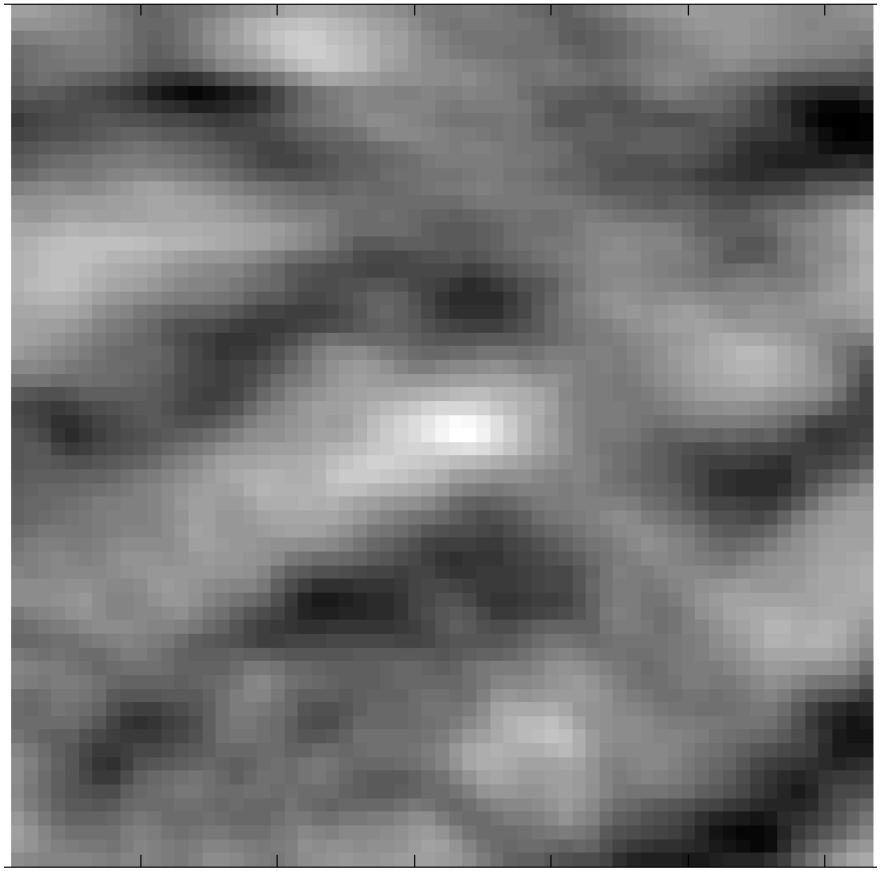
It is EQUIVALENT to the (normalized) SSD:

$$NSSD = \sum_{[i,j]\in R} (\hat{f} - \hat{g})^2 = \sum_{[i,j]\in R} (\hat{f}^2 + \hat{g}^2 - 2\hat{f}\hat{g})$$

$$= 1 + 1 - 2 \sum_{[i,j]\in R} \hat{f}\hat{g} = 2 - 2N_{fg}$$

Normalized Cross Correlation





Highest score coincides with best match. Also, it looks less likely that we could get the wrong match.

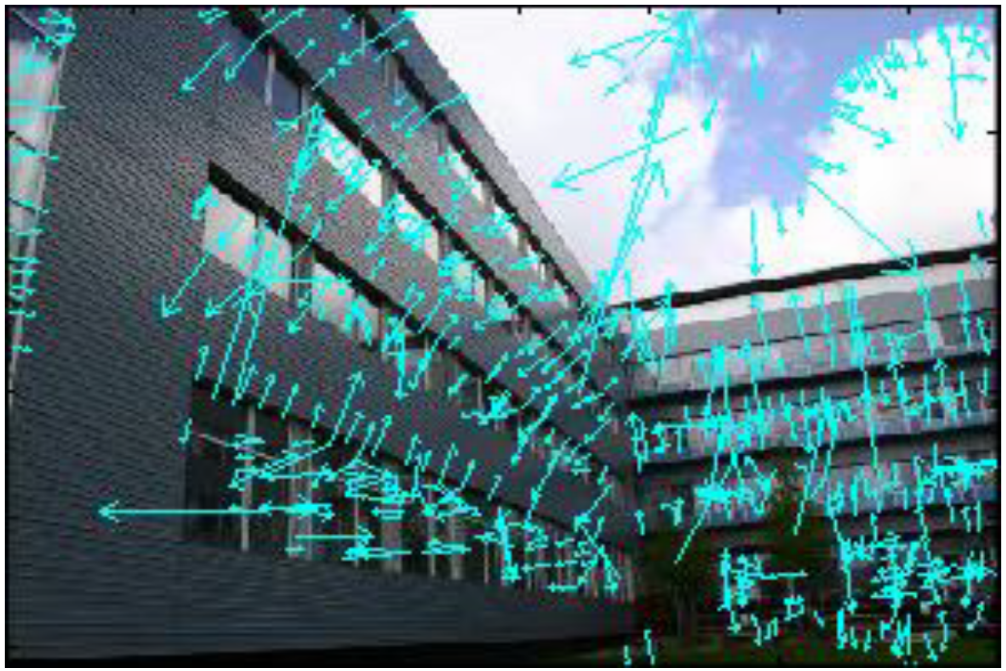
SIFT + RANSAC for Homography with Pure Camera Rotation

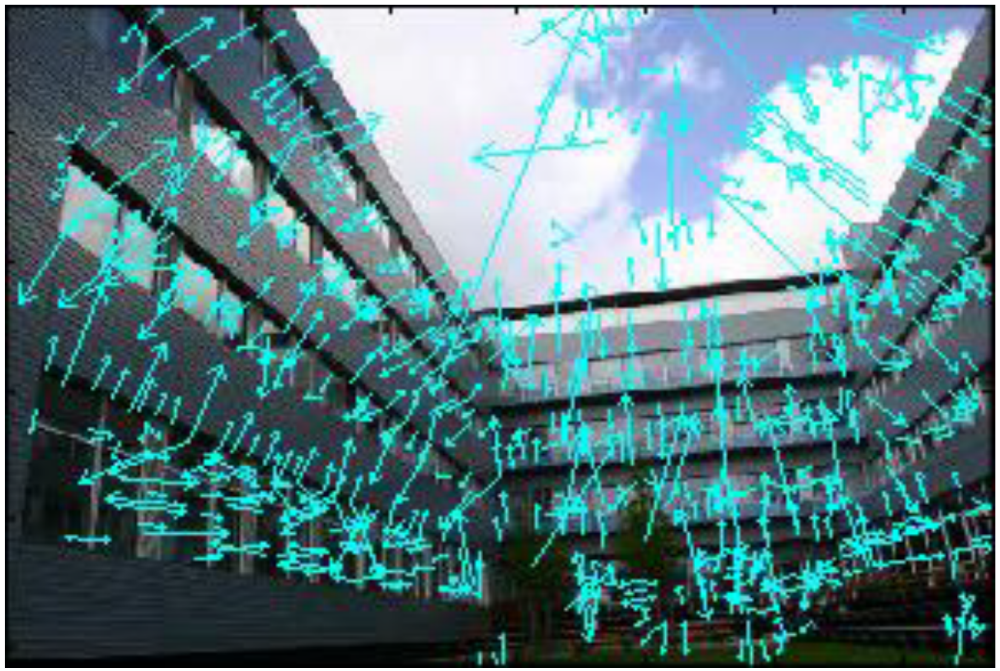
Choose 2 overlapping images.

SIFT + RANSAC for Homography with Pure Camera Rotation

Choose 2 overlapping images.

Find SIFT features for each image.



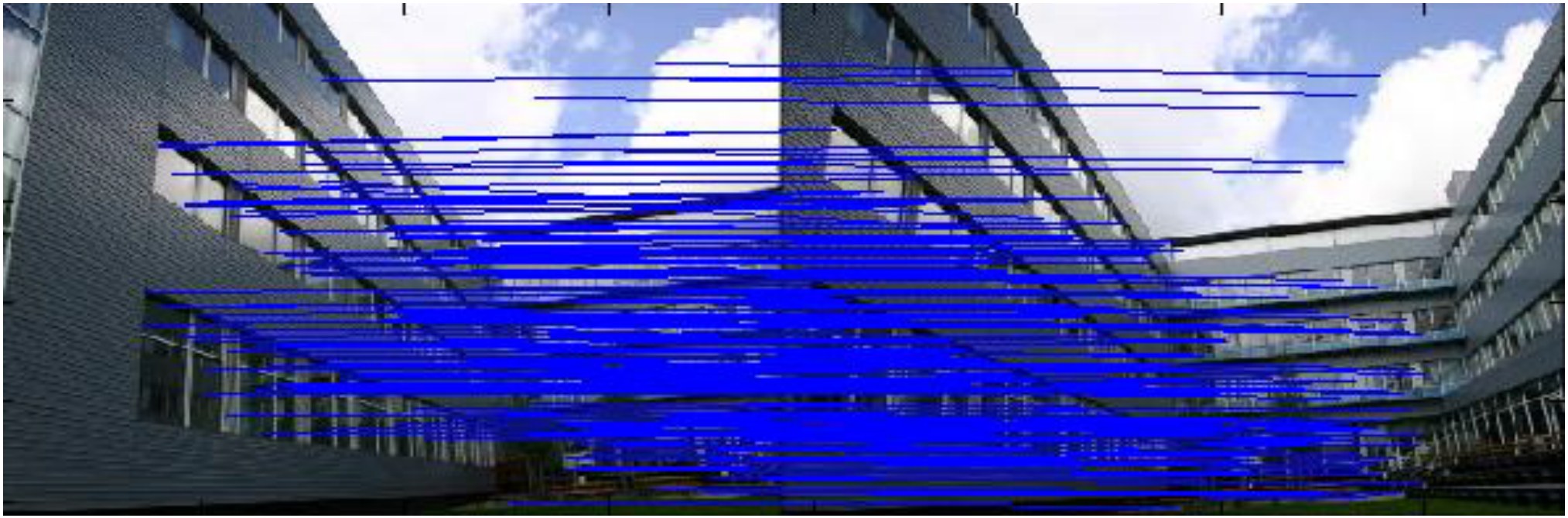


SIFT + RANSAC for Homography with Pure Camera Rotation

Choose 2 overlapping images.

Find SIFT features for each image.

Match SIFT features to get initial point correspondences.



SIFT + RANSAC for Homography with Pure Camera Rotation

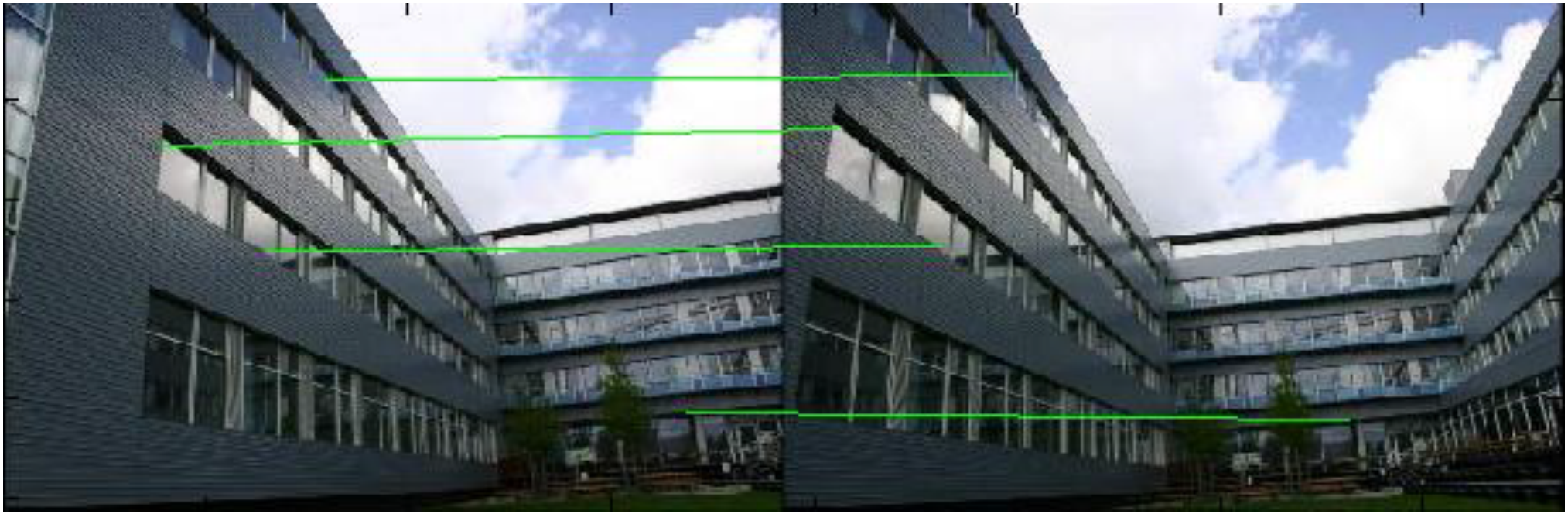
Choose 2 overlapping images.

Find SIFT features for each image.

Match SIFT features to get initial point correspondences.

Run RANSAC:

1. Select minimal number of points (4), find homography.



SIFT + RANSAC for Homography with Pure Camera Rotation

Choose 2 overlapping images.

Find SIFT features for each image.

Match SIFT features to get initial point correspondences.

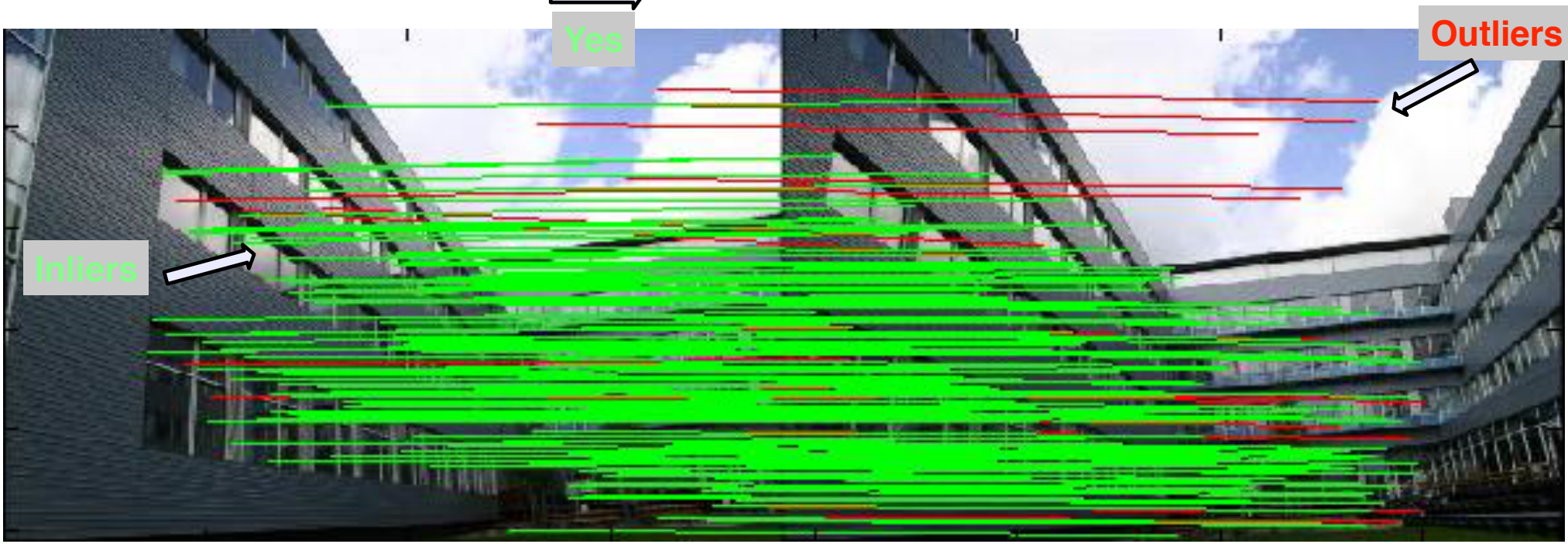
Run RANSAC:

1. Select minimal number of points (4), find homography.
2. Check number of data points consistent with this fit.
3. Good enough?

No

Yes

Find homography using all inliers.



Enforcing 8 DOF

One approach: Set $h_{33} = 1$.

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

Second approach: Impose unit vector constraint

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Subject to the constraint:

$$h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{21}^2 + h_{22}^2 + h_{23}^2 + h_{31}^2 + h_{32}^2 + h_{33}^2 = 1$$

L.S. using Algebraic Distance

Setting $h_{33} = 1$

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

Multiplying through by denominator

$$(h_{31}x + h_{32}y + 1)x' = h_{11}x + h_{12}y + h_{13}$$

$$(h_{31}x + h_{32}y + 1)y' = h_{21}x + h_{22}y + h_{23}$$

Rearrange

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' = x'$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' = y'$$

Algebraic Distance, h_{33}=1 (cont)

2N x 8 **8 x 1** **2N x 1**

Point 1

Point 2

Point 3

Point 4

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

additional points

Algebraic Distance, $h_{33}=1$ (cont)

Linear equations

$$\underset{2Nx8}{\mathbf{A}}\ \underset{8x1}{\mathbf{h}} = \underset{2Nx1}{\mathbf{b}}$$

Solve:

$$\underset{8x2N}{\mathbf{A}^T}\ \underset{2Nx8}{\mathbf{A}}\ \underset{8x1}{\mathbf{h}} = \underset{8x2N}{\mathbf{A}^T}\ \underset{2Nx1}{\mathbf{b}}$$

$$\overbrace{(\mathbf{A}^T\ \mathbf{A})}^{8x8}\ \underset{8x1}{\mathbf{h}} = \overbrace{(\mathbf{A}^T\ \mathbf{b})}^{8x1}$$

$$\mathbf{h} = (\mathbf{A}^T\ \mathbf{A})^{-1}\ (\mathbf{A}^T\ \mathbf{b})$$

Matlab: $\mathbf{h} = \mathbf{A} \backslash \mathbf{b}$

Algebraic Distance, ||h||=1

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Subject to

$$||\mathbf{h}|| = 1$$

Multiplying through by denominator

$$(h_{31}x + h_{32}y + h_{33})x' = h_{11}x + h_{12}y + h_{13}$$

$$(h_{31}x + h_{32}y + h_{33})y' = h_{21}x + h_{22}y + h_{23}$$

Rearrange

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' - h_{33}x' = 0$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' - h_{33}y' = 0$$

Algebraic Distance, ||h||=1 (cont)

2N x 9 **9 x 1** **2N x 1**

4 POINTS

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' & -y_1' \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

additional points

Algebraic Distance, ||h||=1 (cont)

Homogeneous equations

$$\underset{2N\times 9}{\mathbf{A}}\ \underset{9\times 1}{\mathbf{h}} = \underset{2N\times 1}{\mathbf{0}}$$

Solve:

$$\underset{9\times 2N}{\mathbf{A}^T}\ \underset{2N\times 9}{\mathbf{A}}\ \underset{9\times 1}{\mathbf{h}} = \underset{9\times 2N}{\mathbf{A}^T}\ \underset{2N\times 1}{\mathbf{0}}$$

$$\overbrace{(\mathbf{A}^T\ \mathbf{A})}^{9\times 9}\ \underset{9\times 1}{\mathbf{h}} = \underset{9\times 1}{\mathbf{0}}$$

$$\mathbf{SVD\ of\ A^TA} = \mathbf{U}\ \mathbf{D}\ \mathbf{U^T}$$

Let h be the column of U (unit eigenvector) associated with the smallest eigenvalue in D.
(if only 4 points, that eigenvalue will be 0)

Image (un)Warping

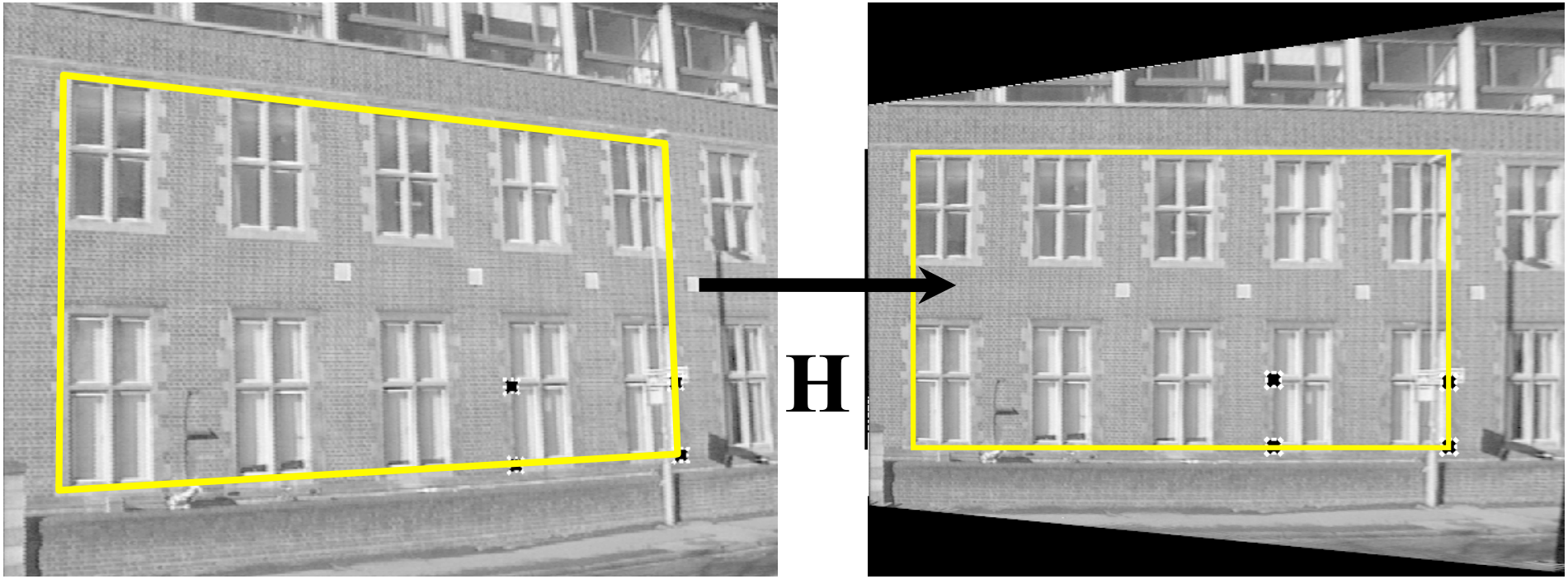
Warping & Bilinear Interpolation

Given an **homography** between two images, (coordinate systems) we want to "warp" one image into the coordinate system of the other.

We will call the coordinate system where we are **mapping from** the "source" image.

We will call the coordinate system we are **mapping to** the "destination" image.

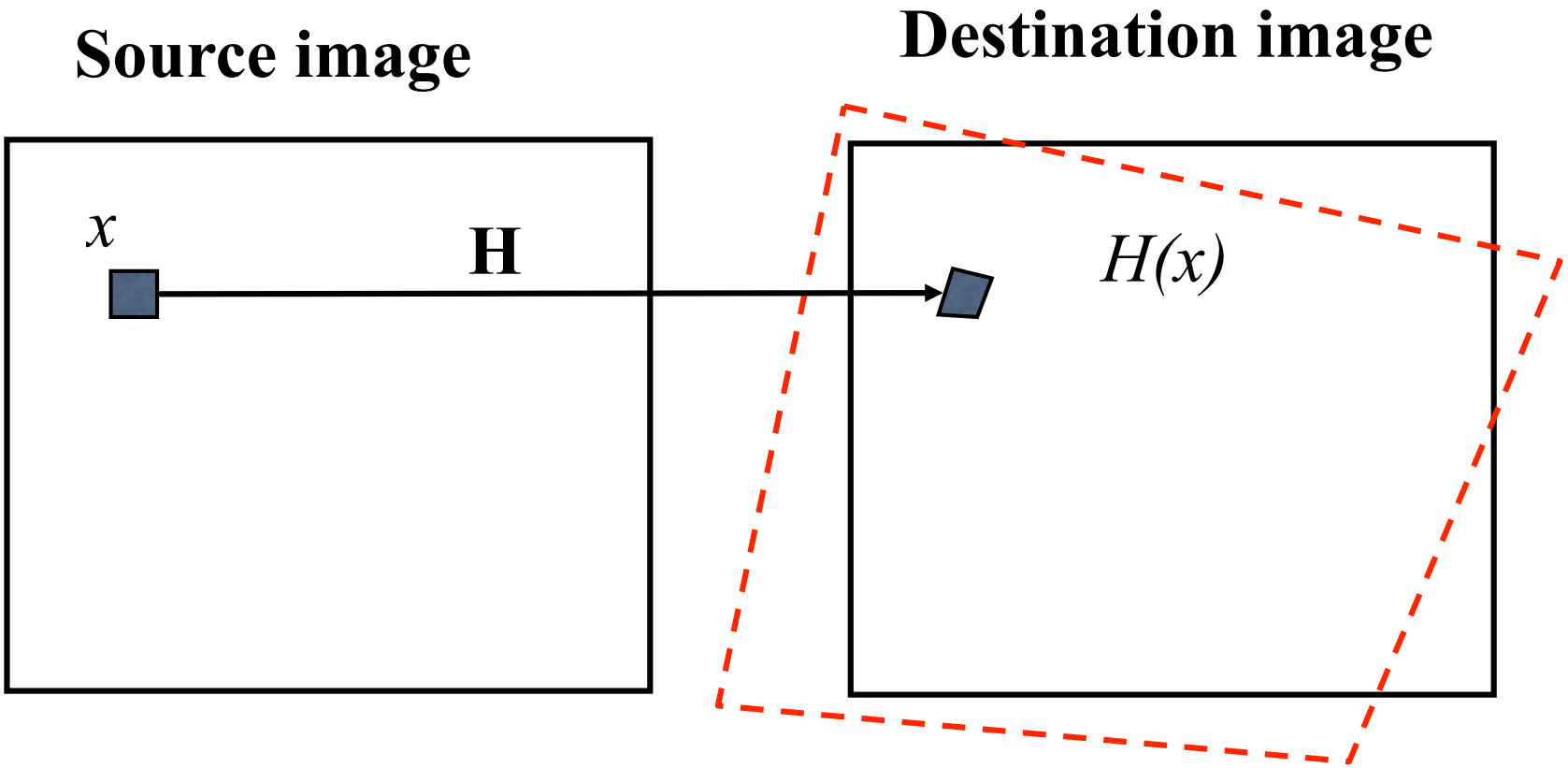
Projective Warping Example



from Hartley & Zisserman

Source Image **Destination image**

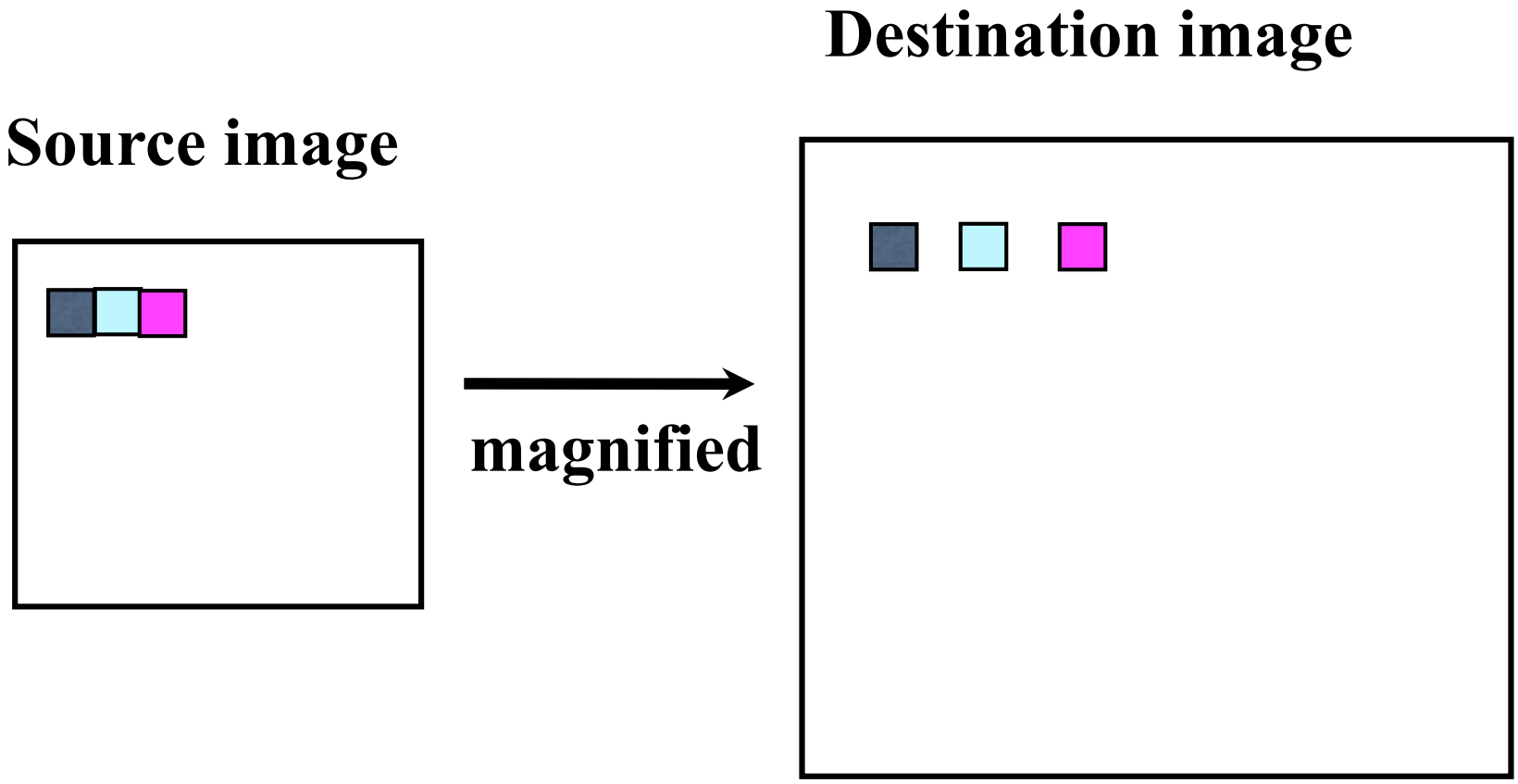
Forward Warping



- For each pixel x in the source image
- Determine where it goes as H(x)
- Color the destination pixel

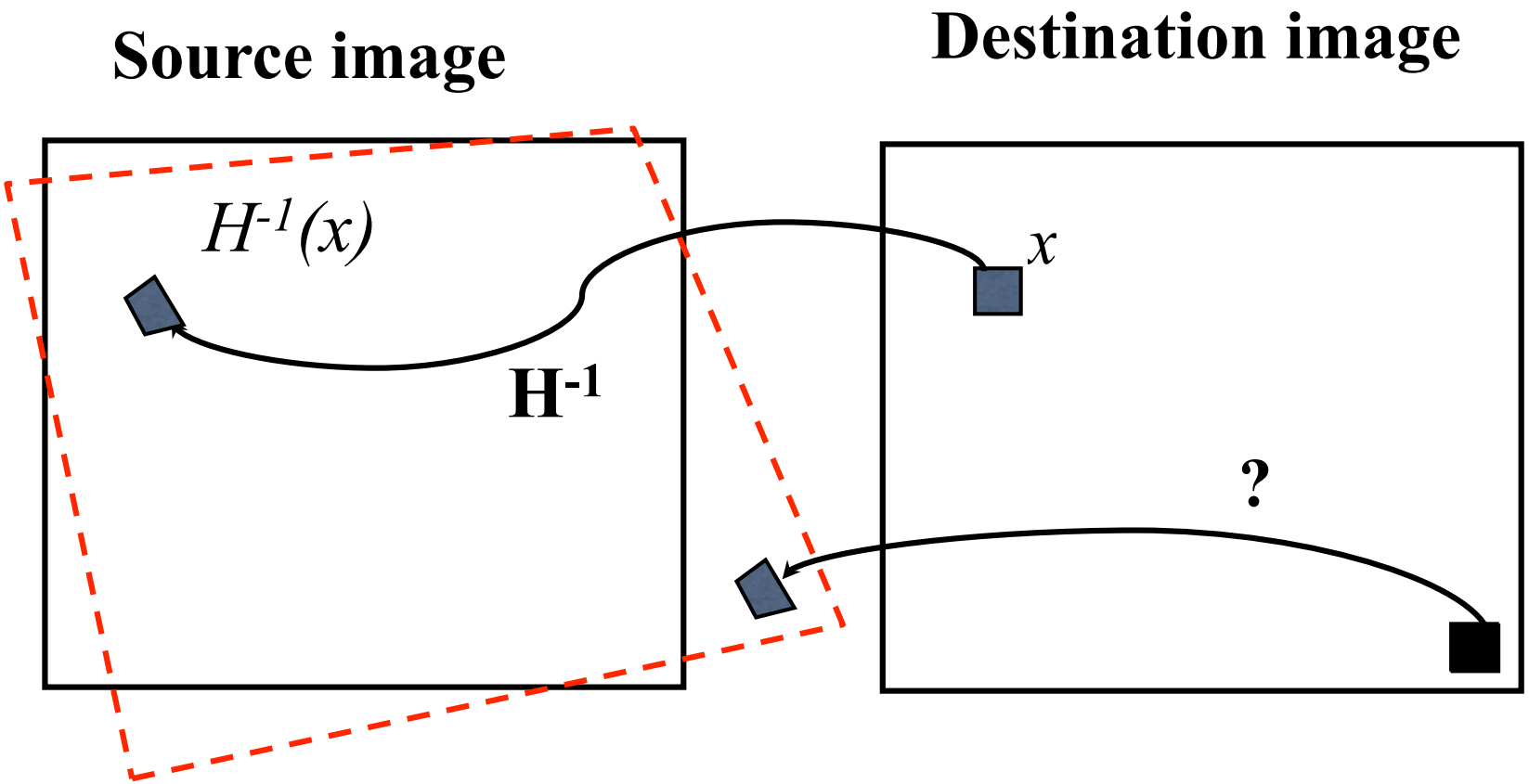
Problems?

Forward Warping Problem



Can leave gaps!

Backward Warping (No gaps)



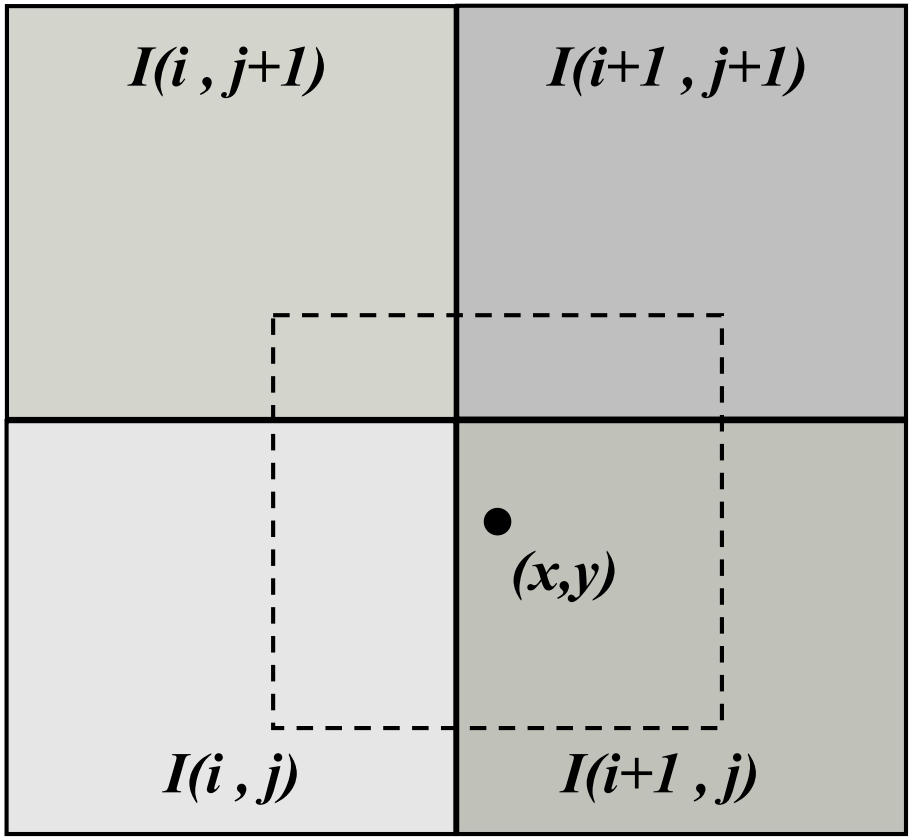
- For each pixel x in the destination image
- Determine where it comes from as $H^{-1}(x)$
- Get color from that location

Interpolation

What do we mean by “get color from that location”?

Consider grey values.

What is the intensity at (x,y)?



Nearest Neighbor:

Take color of pixel with closest center.

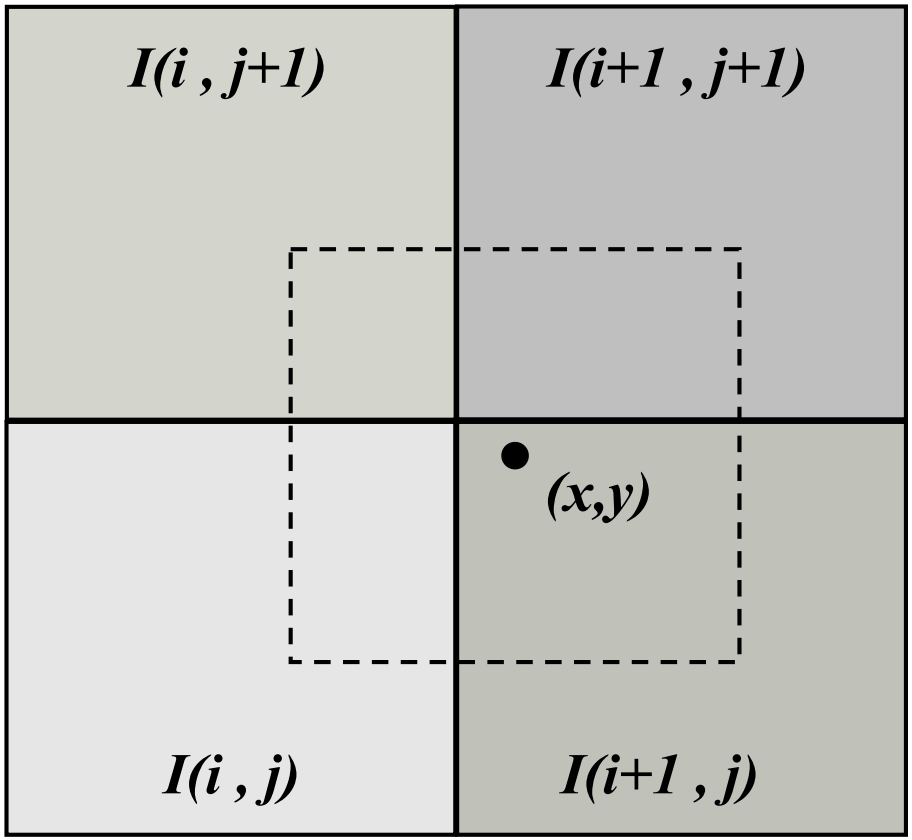
$$I(x,y) = I(i+1,j)$$

Bilinear interpolation

What do we mean by “get color from that location”?

Consider grey values.

What is the intensity at (x,y)?



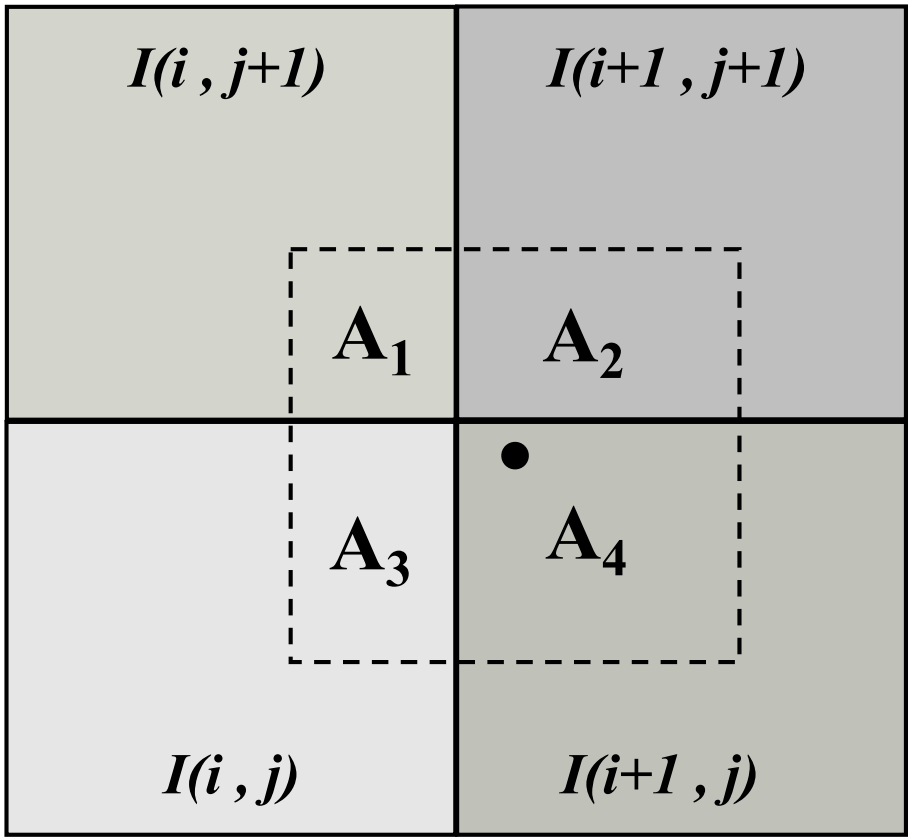
Bilinear Interpolation:

Weighted average

Bilinear interpolation

What do we mean by “get color from that location”?

Consider grey values.



Bilinear Interpolation:

Weighted average

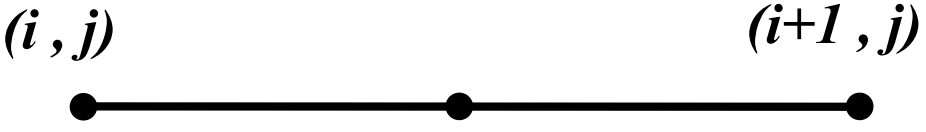
$$I(x,y) = A3*I(i,j) + A4*I(i+1,j) + A2*I(i+1,j+1) + A1*I(i,j+1)$$

Bilinear Interpolation, Math

First, consider linear interpolation

Inituition: Given two pixel values, what should the value be at some intermediate point between them?

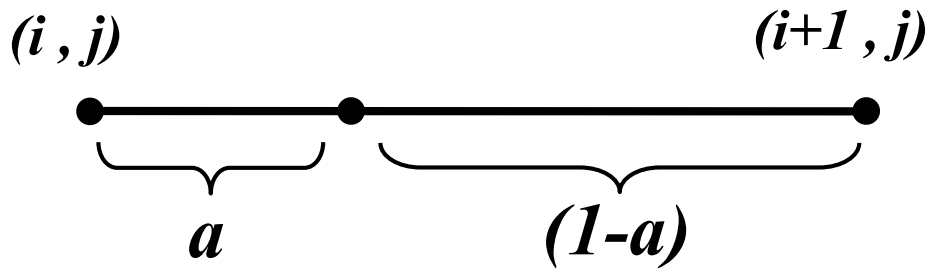
If close to (i,j), should be intensity similar to I(i,j)

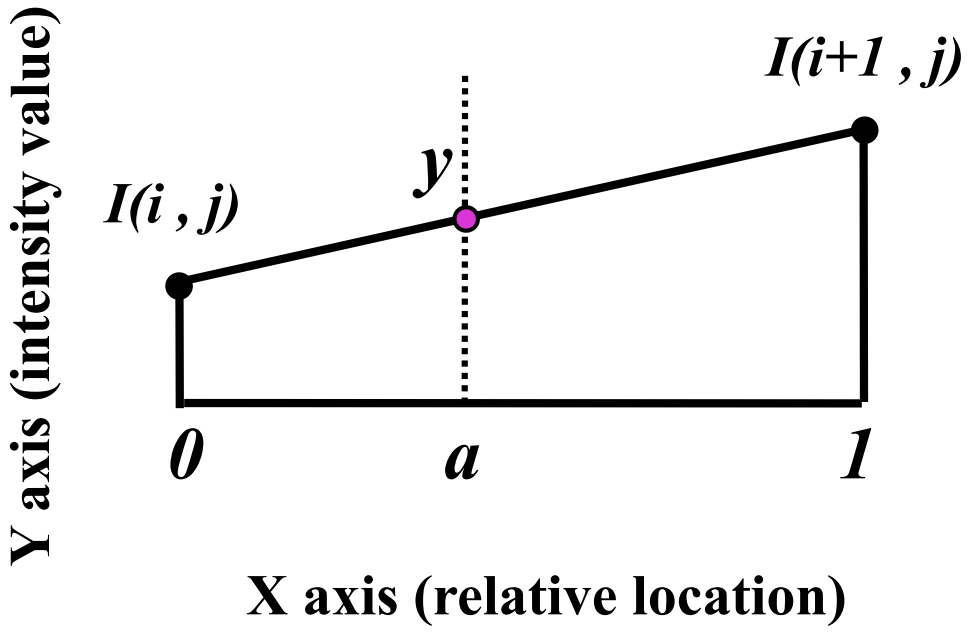


If equidistant from both, should be average of the two intensities

If close to (i+1,j), should be intensity similar to I(i+1,j)

Linear Interpolation





Recall:

$$y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

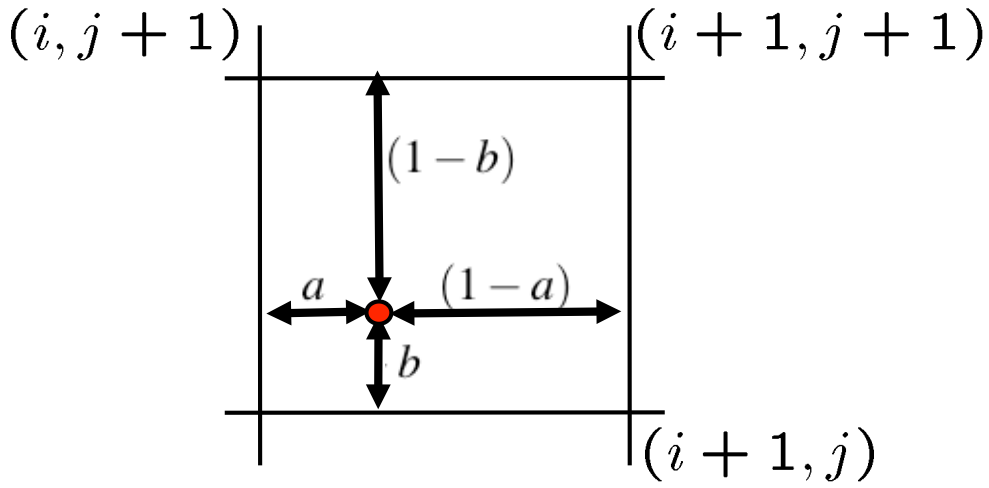
Instantiate

$$y - I(i,j) = \frac{(I(i+1,j) - I(i,j))}{(1-0)}(a-0)$$

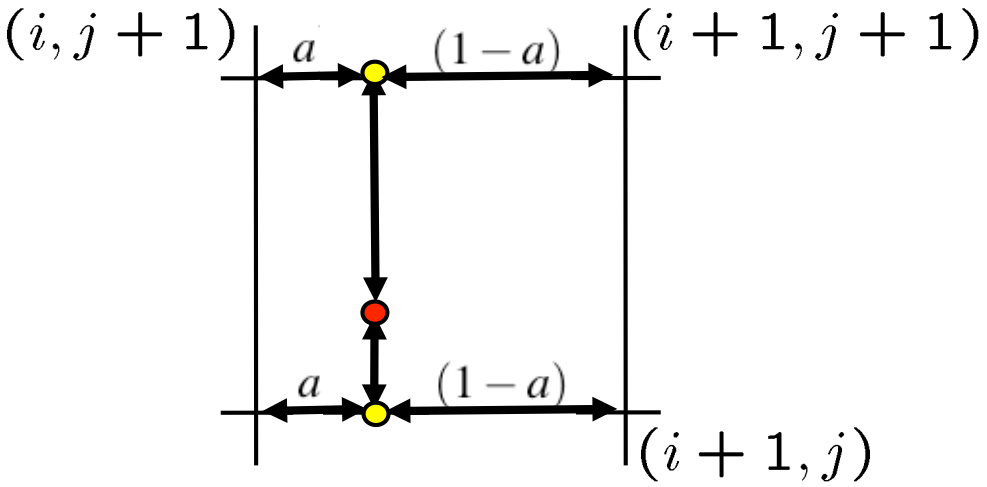
Solve

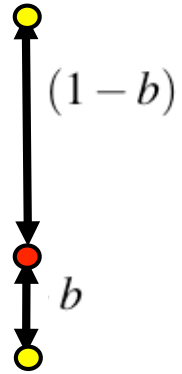
$$y = (1-a)\, I(i,j) + a\, I(i+1,j)$$

Bilinear Interpolation, Math



Bilinear Interpolation, Math

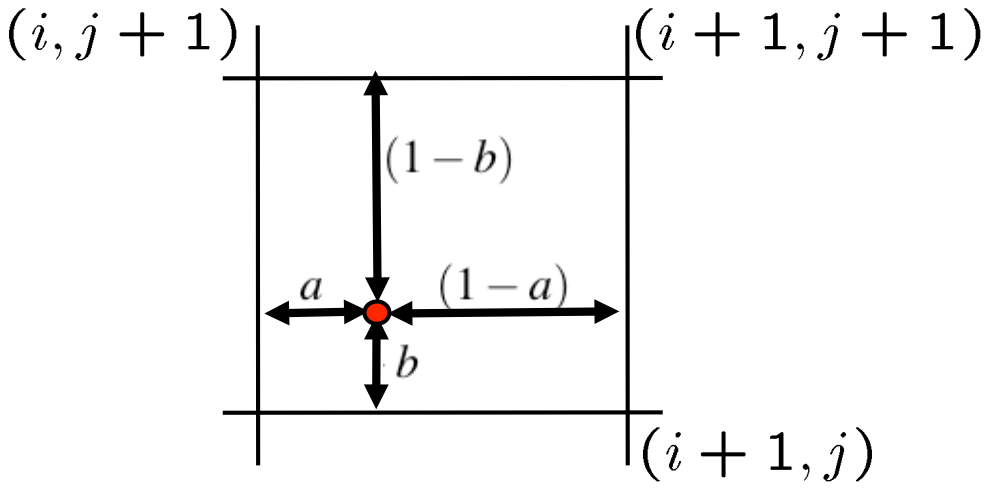




$$(1-b)\,[\,(1-a)\,I(i,j)+a\;I(i+1,j)\,]$$

$$+\;\;b\;\,[\,(1-a)\,I(i,j+1)+a\;I(i+1,j+1)\,]$$

Bilinear Interpolation, Math

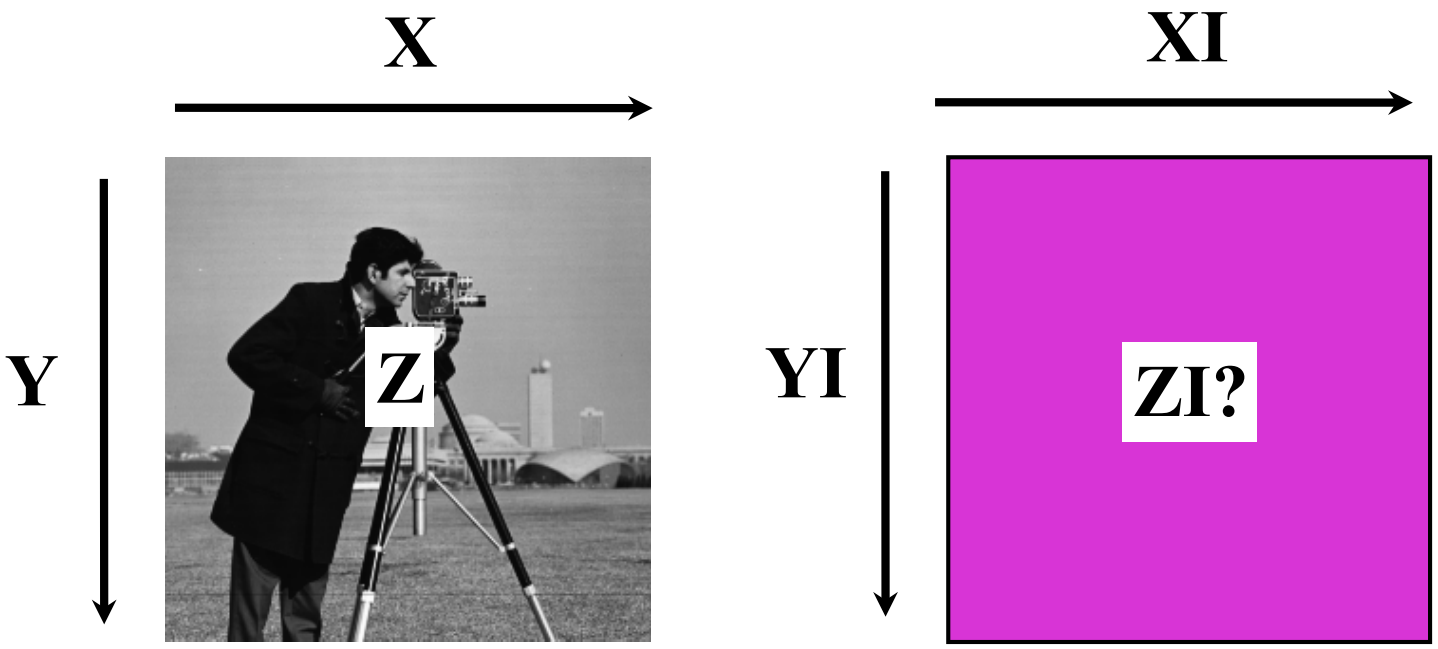


$$\begin{aligned}\mathbf{I} = \quad & (1-a)(1-b)\ I(i,j) \\ & +a\ (1-b)\ I(i+1,j) \\ & +(1-a)\ b\ I(i,j+1) \\ & +a\ b\ I(i+1,j+1)\end{aligned}$$

Image Warping in Matlab

`interp2` is Matlab's built-in function for image warping

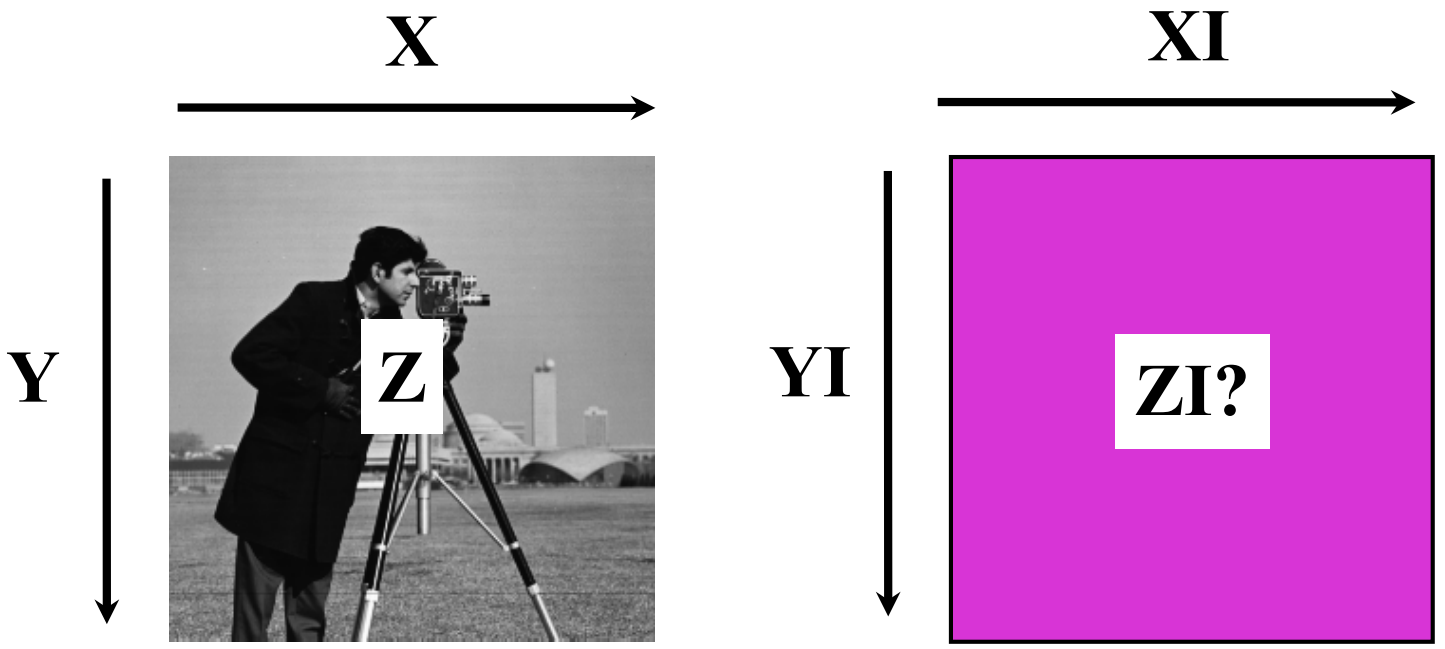
Usage: `interp2(X,Y,Z,XI,YI)`



Tips on Using Interp2

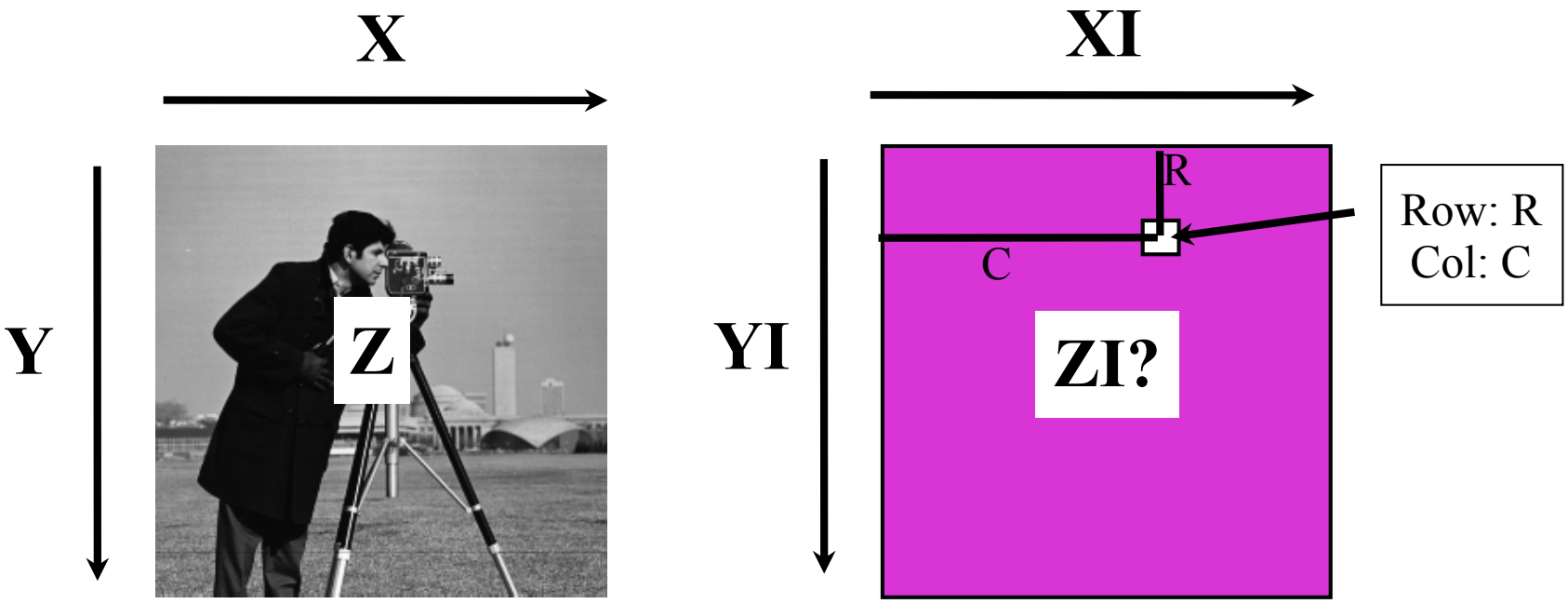
For our purposes, we can assume X and Y are just normal image pixel coords.

Simpler Usage: `interp2(Z,XI,YI)`



Tips on Using Interp2 (cont)

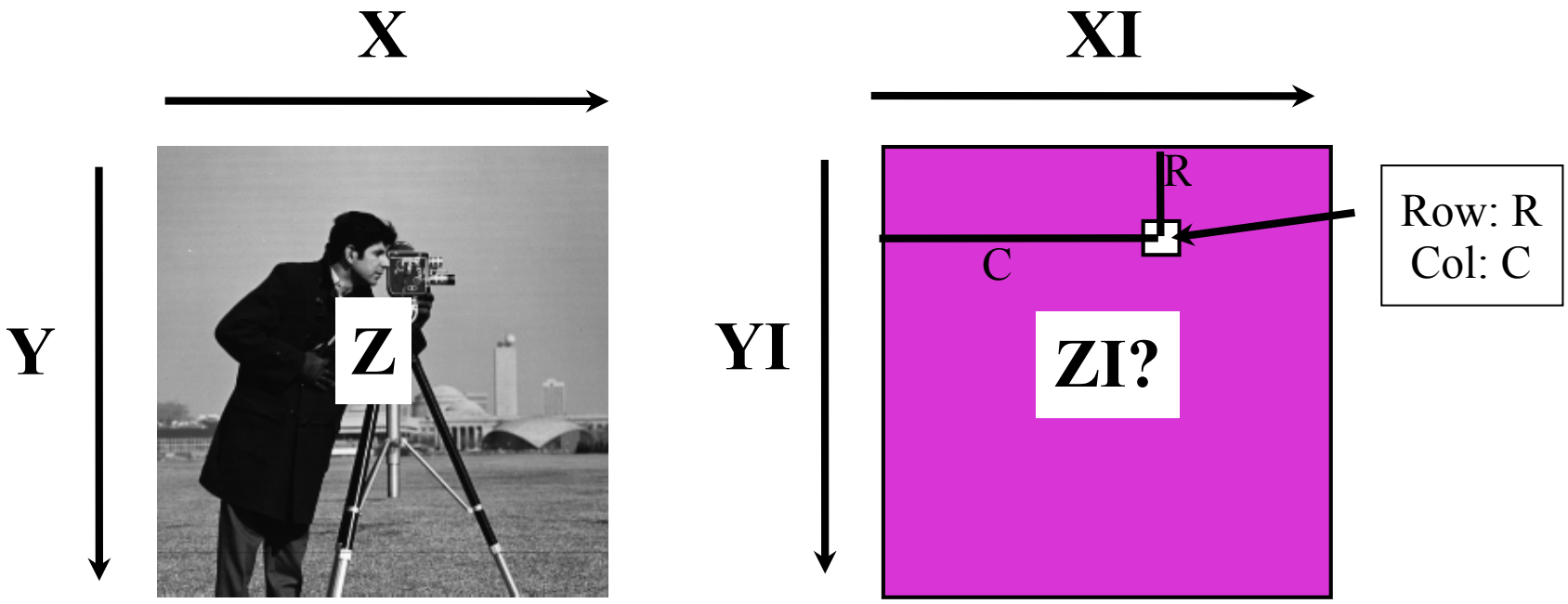
XI and YI are two arrays the same size as ZI.
For a given row and col (R,C), the value (XI(R,C), YI(R,C)) tells which (X,Y) coord of the orig image to take.
That is: Z(YI(R,C), XI(R,C)).



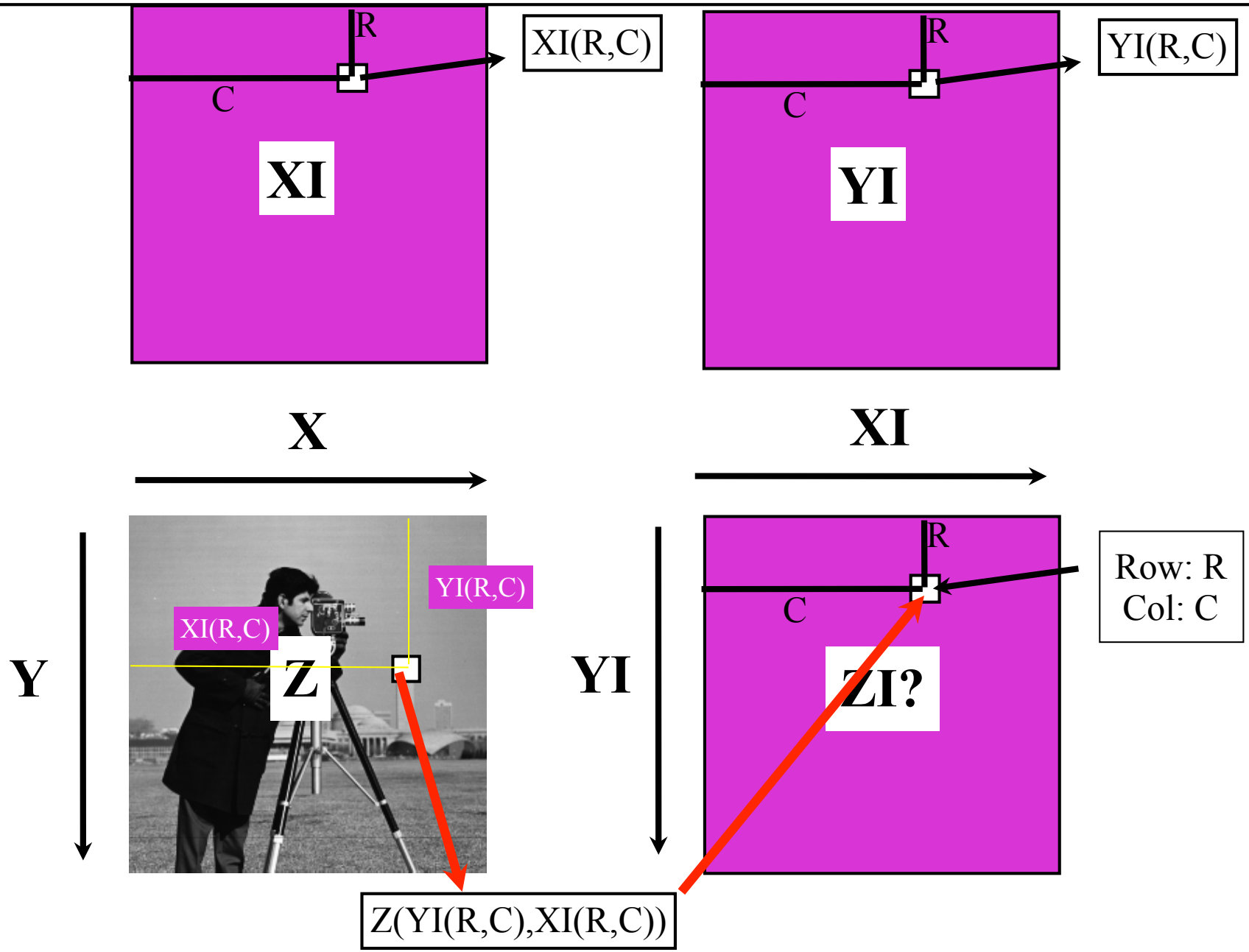
Tips on Using Interp2 (cont)

How does it work?

Consider computing the value of ZI at row R and col C.



Tips on Using Interp2 (cont)



Tips on Using Interp2 (cont)

`interp2` takes care of bilinear interpolation for you, in case YI(R,C) and XI(R,C) are not integer coords.

There are optional arguments to **`interp2`** to change the way the interpolation is calculated. We can go with the **default**, which is bilinear interp.

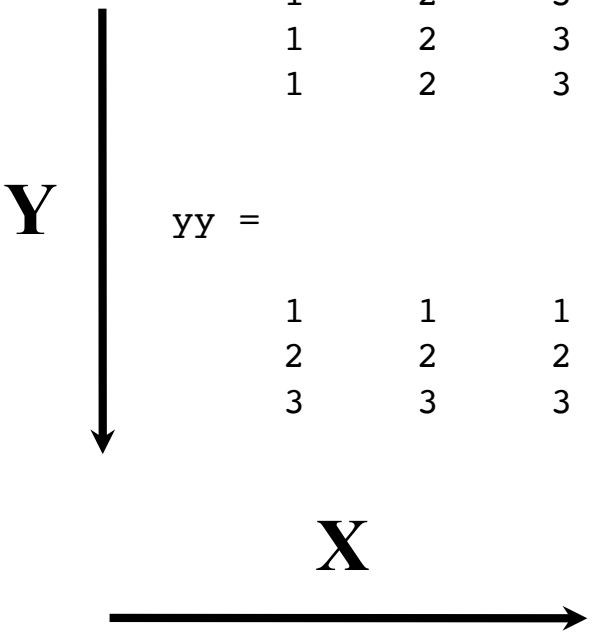
Meshgrid

A useful function when using `interp2` is `meshgrid`

```
>> [xx,yy] = meshgrid(1:4,1:3)

xx =

     1     2     3     4
     1     2     3     4
     1     2     3     4


yy =

     1     1     1     1
     2     2     2     2
     3     3     3     3
```

An array suitable for use as XI

An array suitable for use as YI



Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, xi, yi);
imshow(uint8(foo));
```

Result: just a copy of the image.

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, xi/2, yi/2);
imshow(uint8(foo));
```

Result: scale up by 2
(but stuck in 256x256 image)

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, 2*xi, 2*yi);
imshow(uint8(foo));
```

Result: scale down by 2
(within 256x256 image, pad with 0)

Interp2 Examples

Confusion alert!

```
foo = interp2(im, xi/2, yi/2);
```

Divide coords by 2 to scale up by 2

```
foo = interp2(im, 2*xi, 2*yi);
```

Multiply coords by 2 to reduce up by 2

`interp2` needs the **inverse coordinate transforms** to the geometric operation you want (it uses backward warping)

Interp2 Examples

A more complicated example: scale down by 2, but around center of image (128,128), not (0,0)

Recall concatenation of transformation matrices:

$$H = \begin{bmatrix} 1 & 0 & 128 \\ 0 & 1 & 128 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -128 \\ 0 & 1 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

Bring origin back to (128,128)

Scale down by 2 around origin

Bring (128,128) to origin

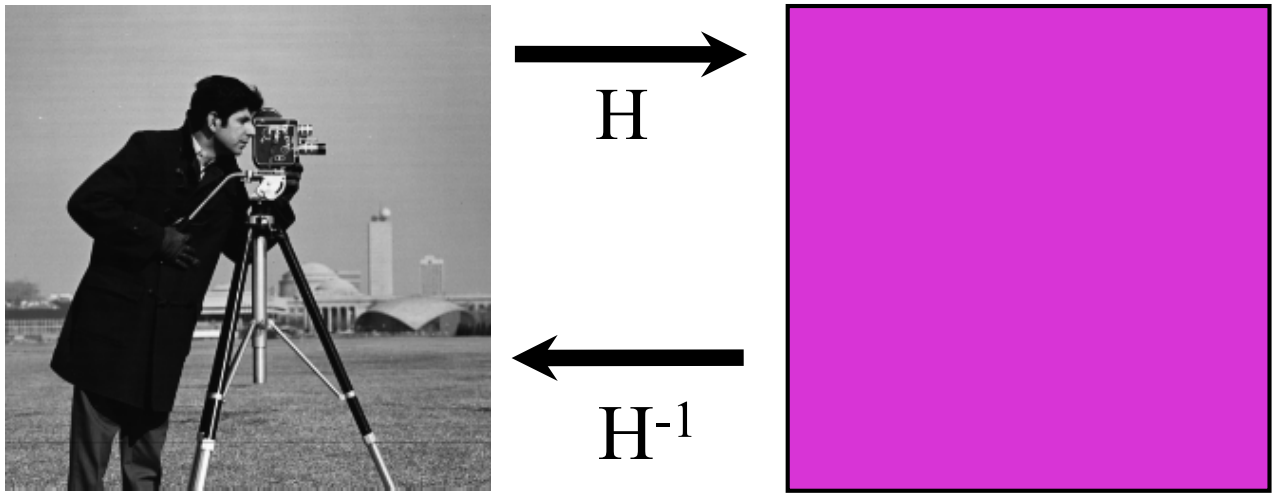
$$H = \begin{bmatrix} .5 & 0 & 64 \\ 0 & .5 & 64 \\ 0 & 0 & 1 \end{bmatrix}$$

BUT, BE CAREFUL….

Interp2 Examples

$$H = \begin{bmatrix} .5 & 0 & 64 \\ 0 & .5 & 64 \\ 0 & 0 & 1 \end{bmatrix}$$

This specifies how we want the original image to map into the new image.



`interp2` needs to know how the new image coords map back to the original image coords.

$$H^{-1} = \begin{bmatrix} 2 & 0 & -128 \\ 0 & 2 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

Interp2 Examples

$$H = \begin{bmatrix} 2 & 0 & -128 \\ 0 & 2 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, 2*xi-128, 2*yi-128);
imshow(uint8(foo));
```

Result

Interp2 Examples

More generally (e.g. for using homographies)

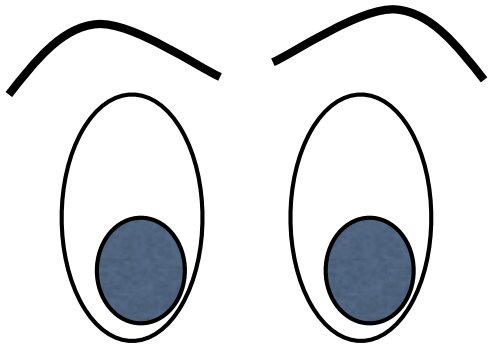
$$H = \begin{bmatrix} 1 & 0 & 128 \\ 0 & 1 & 128 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -128 \\ 0 & 1 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = H^{-1}$$

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

h = [1 0 128;0 1 128; 0 0 1] * [1/2 0 0; 0 1/2 0; 0 0 1] * [1 0 -128; 0 1 -128; 0 0 1];
h = inv(h);   %TAKE INVERSE FOR USE WITH INTERP2
xx = (h(1,1)*xi+h(1,2)*yi+h(1,3))./(h(3,1)*xi+h(3,2)*yi+h(3,3));
yy = (h(2,1)*xi+h(2,2)*yi+h(2,3))./(h(3,1)*xi+h(3,2)*yi+h(3,3));
foo = uint8(interp2(im,xx,yy));
figure(1); imshow(foo)
```

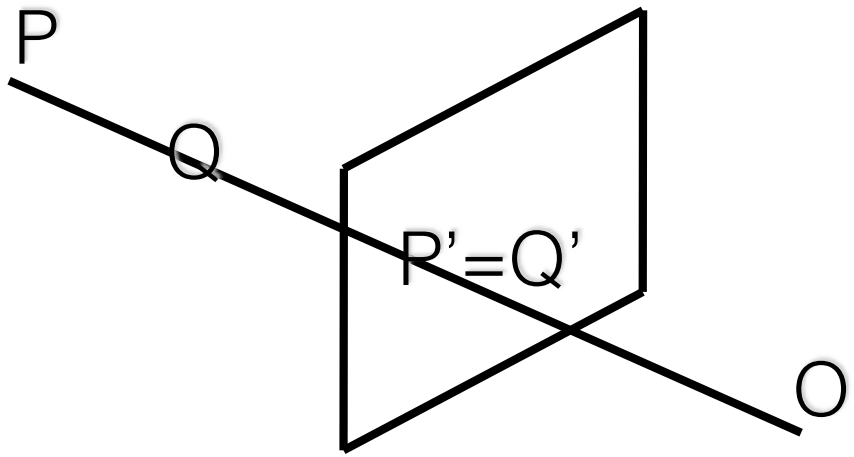
Result

Stereo Vision



Why Stereo Vision?

2D images project 3D points into 2D:



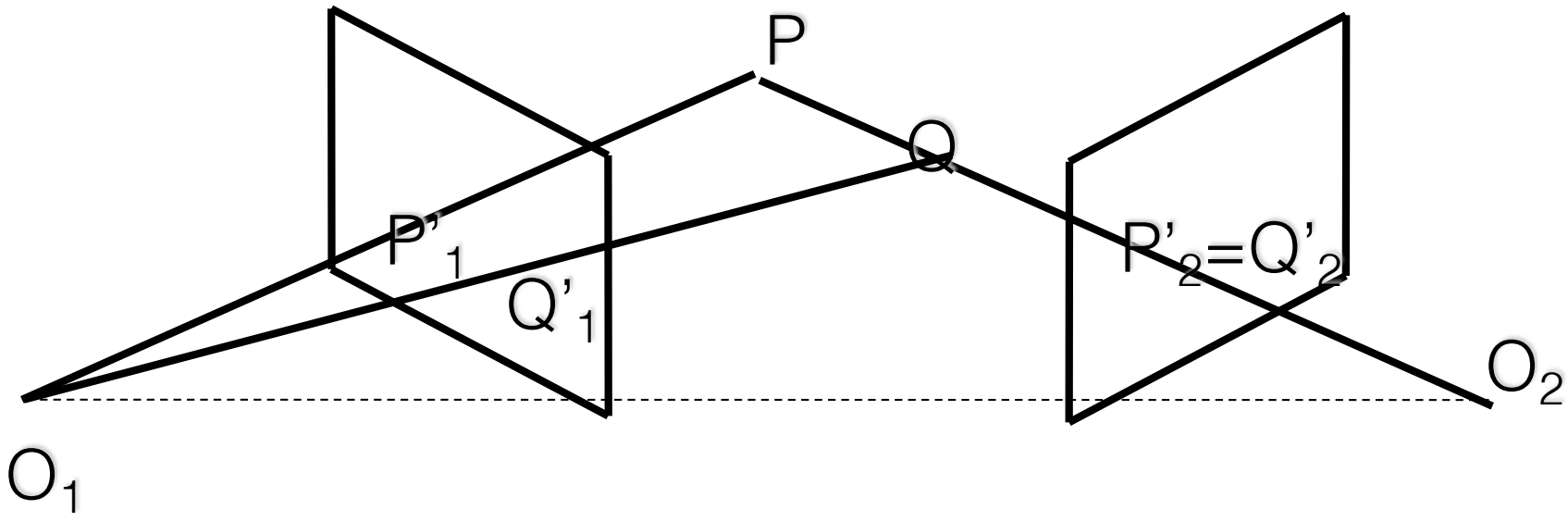
- 3D Points on the same viewing line have the same 2D image:
- 2D imaging results in depth information loss

Stereo Vision

Refers to the ability of:

The ability to infer information on the 3D structure and distance of a scene from two or more images taken from different viewpoints.

Recovering Depth Information:



Depth can be recovered with two images and triangulation.

Stereo Vision Problems:

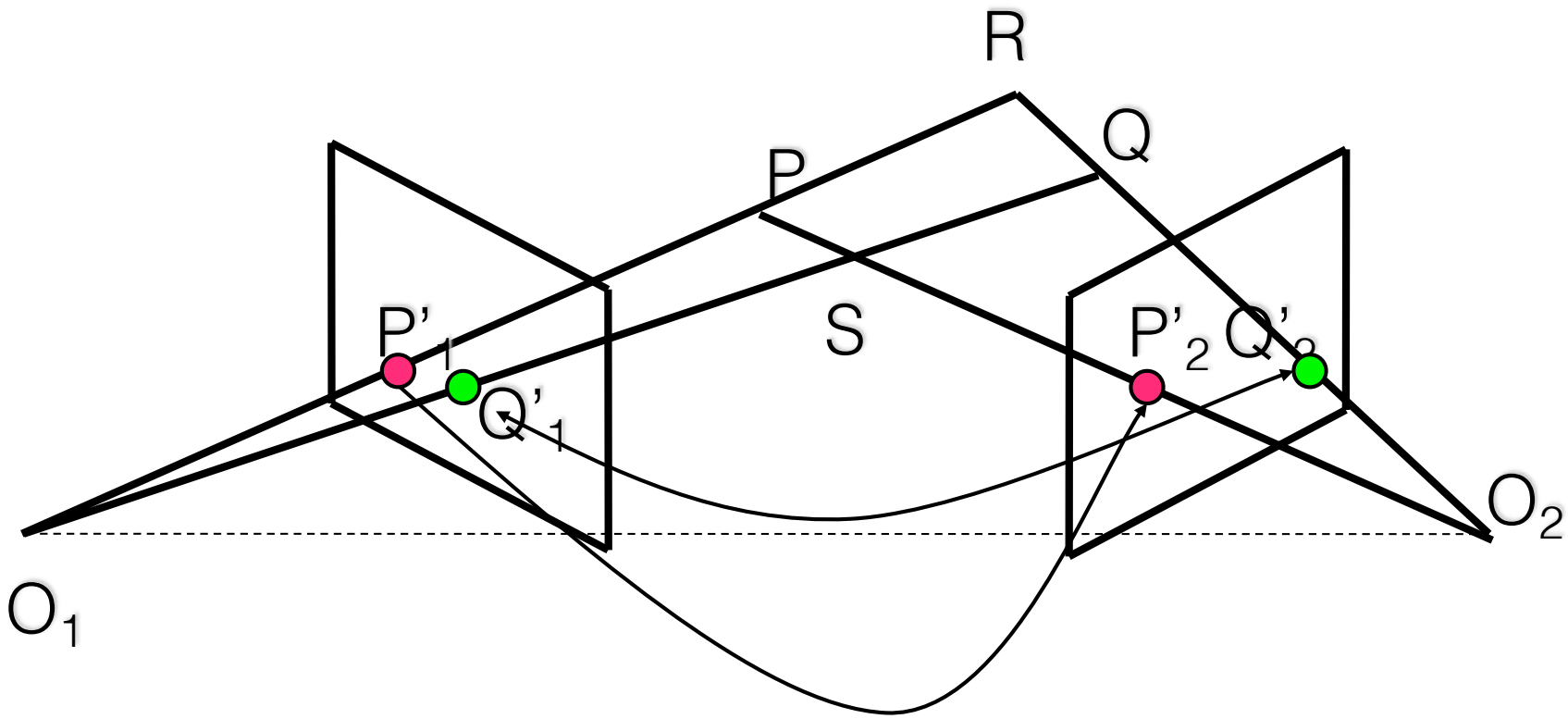
Correspondence Problem:

Determining which pixel on the left corresponds to which pixel on the right.

Reconstruction Problem:

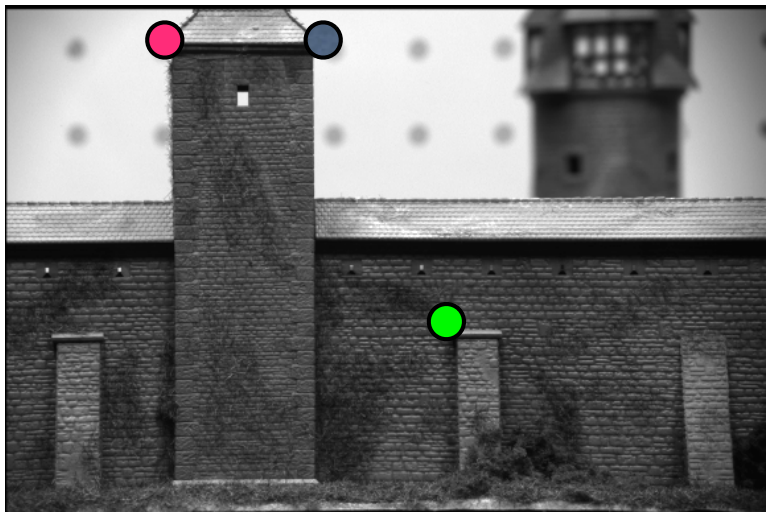
Given a number of correspondence pairs and camera geometry information, find location and 3D structure of the observed objects.

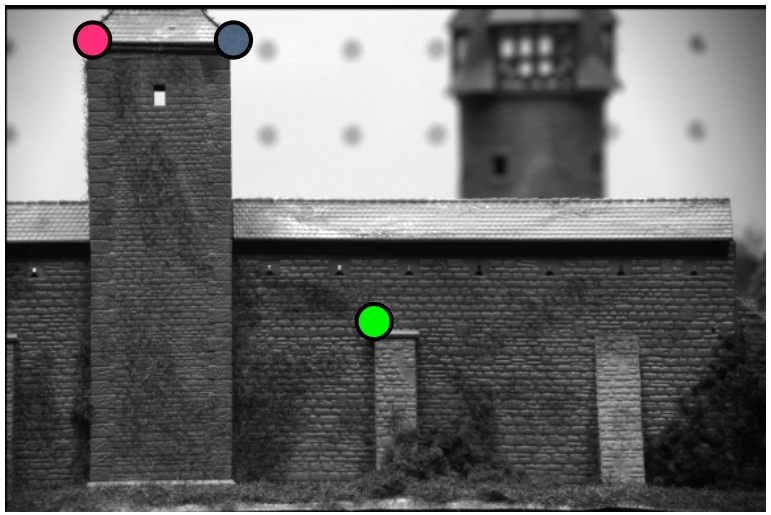
Finding Correspondences:



Wrong correspondences can result in large depth errors!

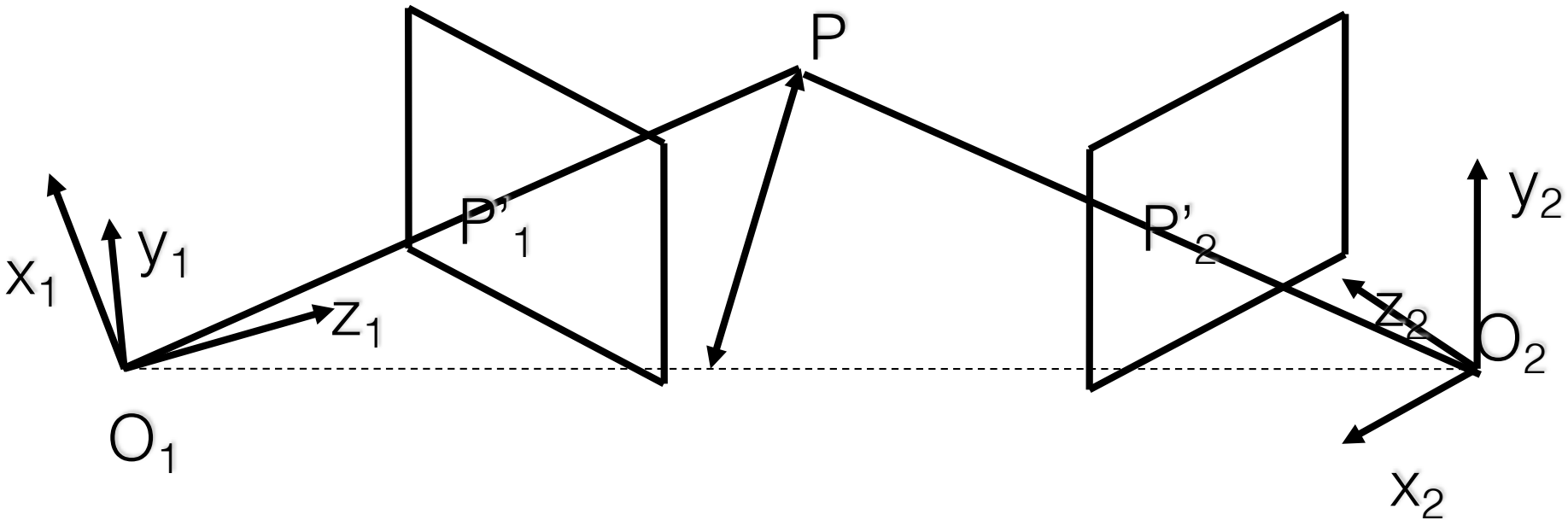
Finding Correspondences:





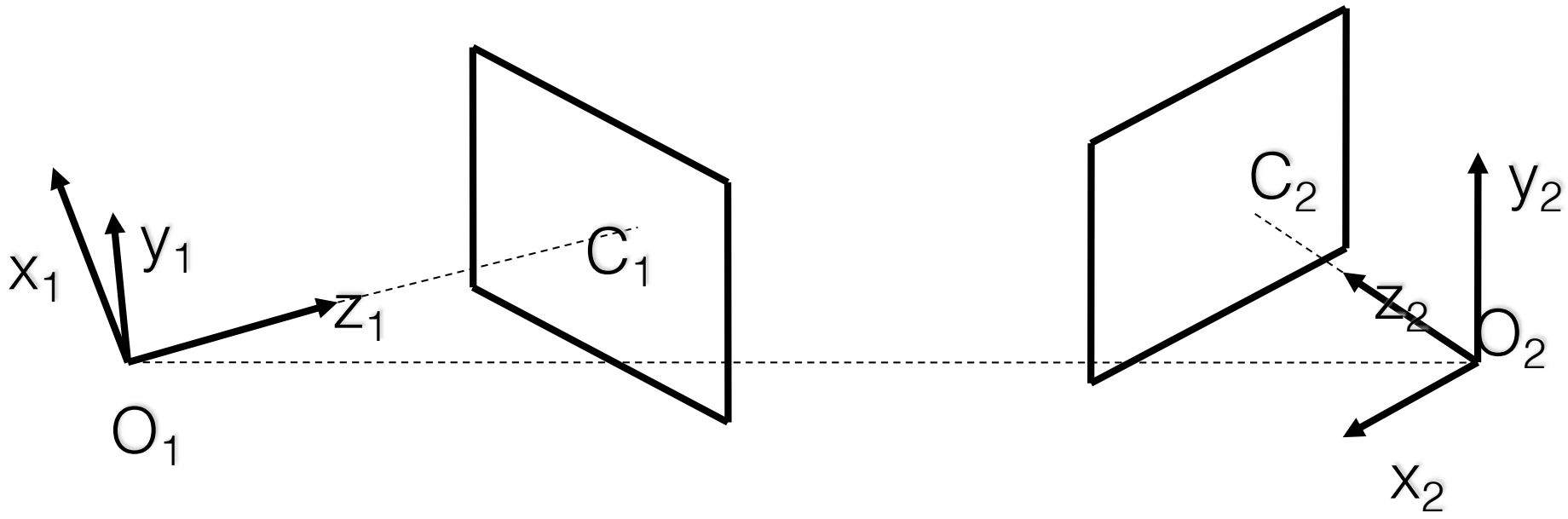
3D Reconstruction

Given the correspondences, “triangulate” to find depth:



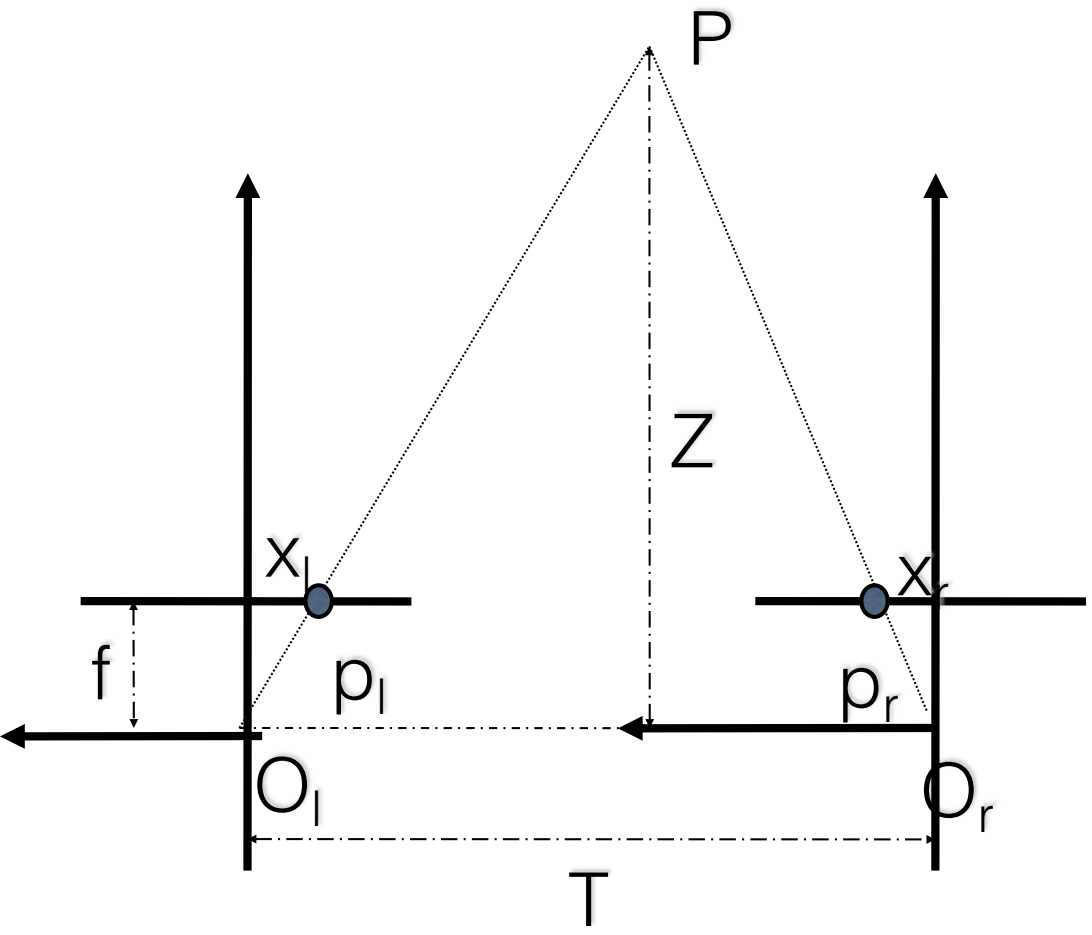
The stereo rig must be calibrated!

Parameters of a Stereo System



- Intrinsic:
  - f_1 and f_2: focal lengths
  - c_1 and c_2: principal points
  - Pixel size
- Extrinsic
  - Transformation (R,T) between cameras

A simple stereo system



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

$$Z = f\frac{T}{x_r - x_l}$$

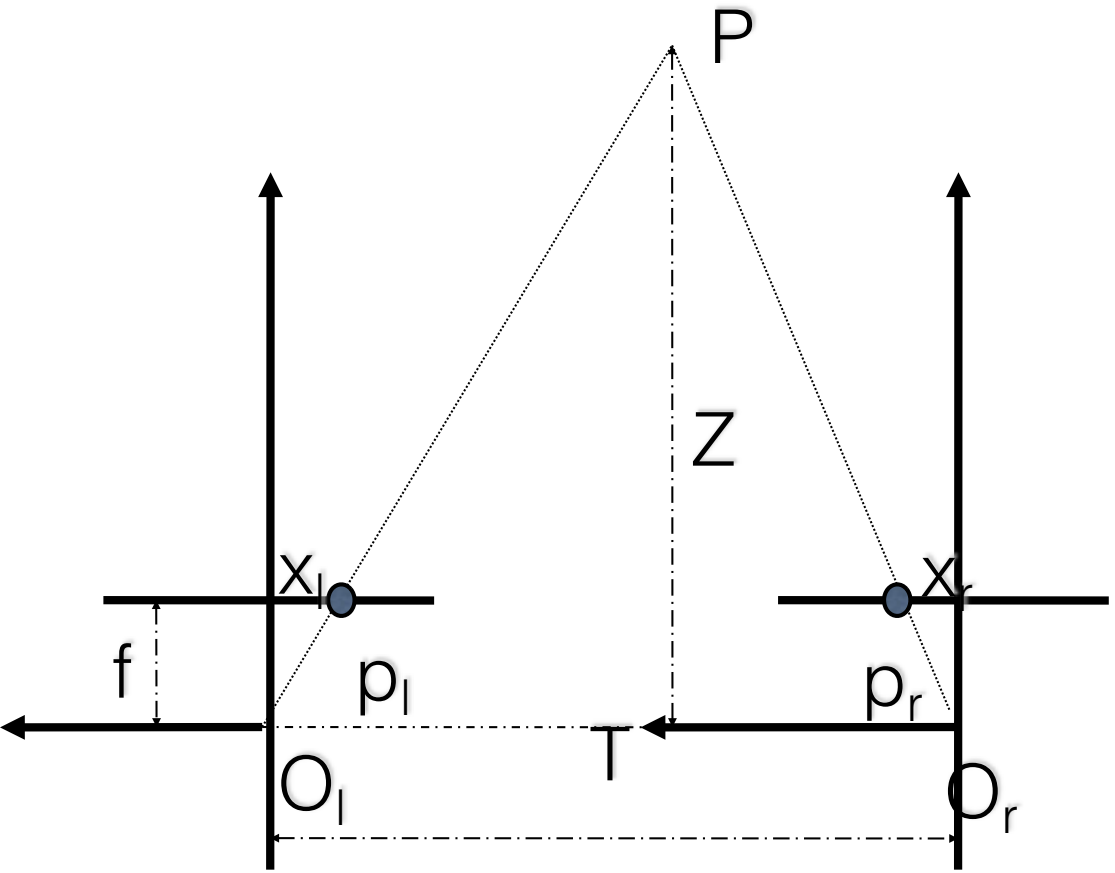
$$\boxed{Z = f\frac{T}{d}}$$

Disparity: $d = x_r - x_l$

T is the stereo baseline

d measures the difference in retinal position between corresponding points

A simple stereo system



$$Z = f\frac{T}{d}$$

- Depth is inversely proportional to disparity

The Correspondence Problem

Basic assumptions:

Most scene points are visible in both images

Corresponding image regions are similar

These assumptions hold if:

The distance of the fixation point from the cameras is much larger than the stereo baseline: $Z >> T$

The Correspondence Problem

Is a "search" problem:

Given an element in the left image, search for the corresponding element in the right image.

We must choose:

Elements to match

A similarity measure to compare elements

Correspondence Problem

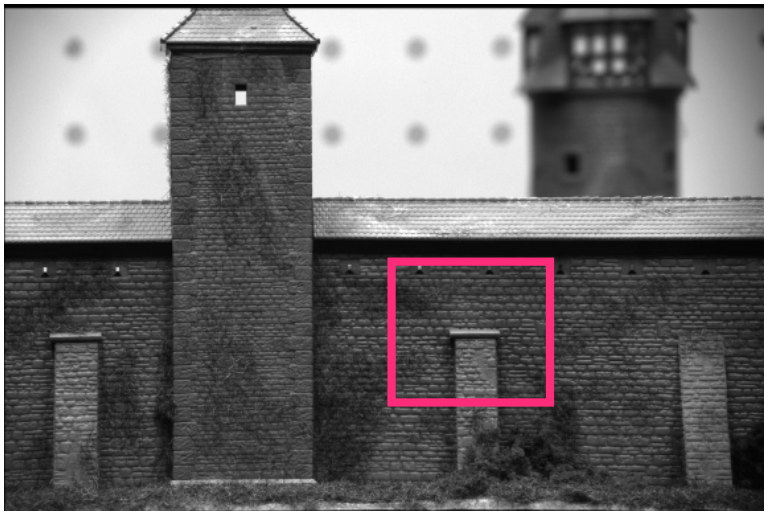
Two classes of algorithms:

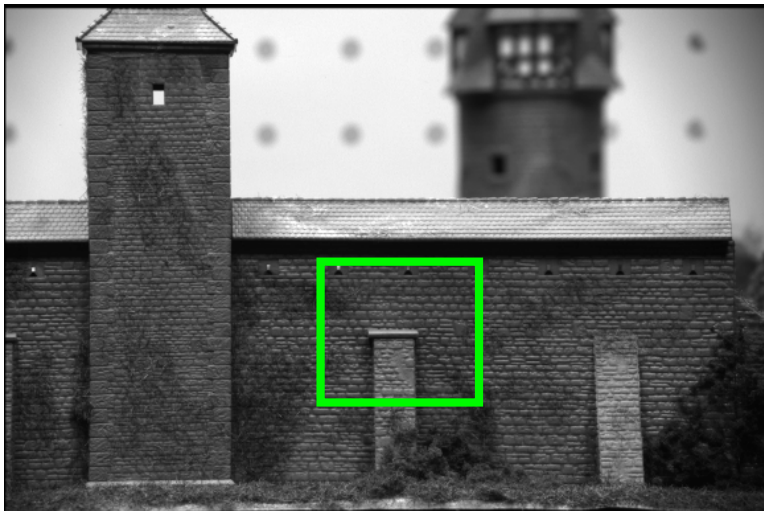
- Correlation-based algorithms
  - Produce a DENSE set of correspondences
- Feature-based algorithms
  - Produce a SPARSE set of correspondences

Correlation-based Algorithms

Elements to be matched:

image WINDOWS of fixed size.





Finding the disparity map

Inputs:

- Left image I_l
- Right image I_r

Parameters that must be chosen:

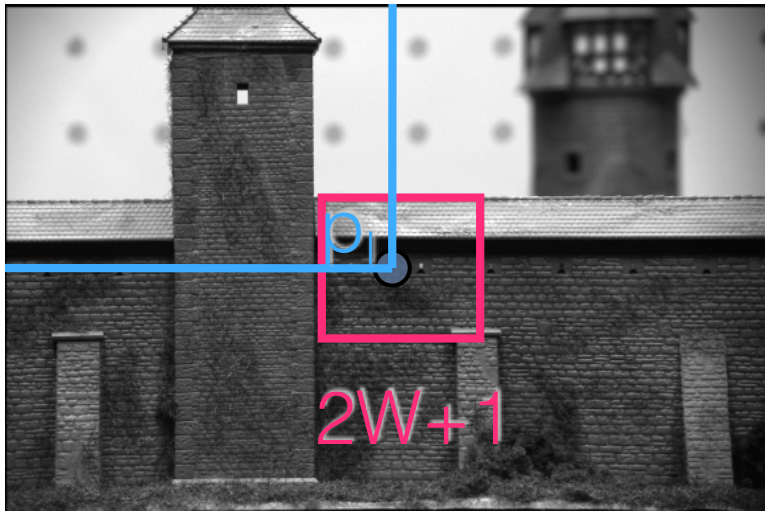
- Correlation Window size 2W+1
- Search Window size ω
- Similarity measure Ψ

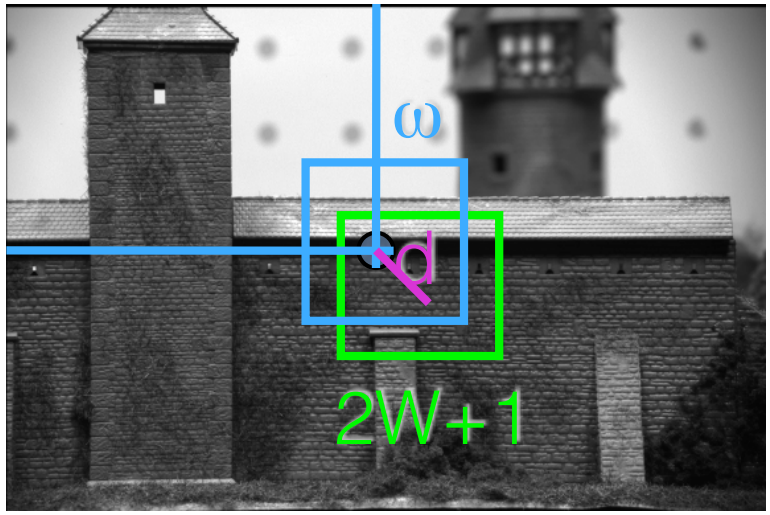
CORR_MATCHING Algorithm

Let p_l and p_r be pixels on the I_l and I_r

Let $R(p_l)$ be the search window ω x ω on I_r associated with p_l

Let d be the displacement between p_l and a point in $R(p_l)$.





CORR_MATCHING Algorithm

For each pixel $p_l=[i,j]$ in I_l do:

For each displacement $d=[d_1,d_2]$ in $R(p_l)$ compute:

$$C(d) = \sum_{l=-W}^{l=W} \sum_{k=-W}^{k=W} \Psi(I_l(i+k,j+l),I_r(i+k-d_1,j+l-d_2))$$

The disparity at p_l is the vector d with best C(d) over $R(p_l)$

(i.e. max. C_{fg}, or min. SSD)

Output the disparity for each pixel p_l

How do we set W, R and ω ?

W (width of the correlation window):

should be based on the "scale" of the scene.

R(P_l) and ω (search window):

Should be estimated based on the range of scene distances and the baseline:

Z = fT/d or d = fT/Z

Feature-based Methods

Similar to Correlation-based methods, but:

- They only search for correspondences of a **sparse** set of image features.
- Correspondences are given by the most similar feature pairs.
- Similarity measure must be adapted to the type of feature used.

Feature-based Methods:

Features most commonly used:

- Corners, SIFT
  - Similarity measured in terms of:
    - surrounding gray values (SSD, Cross-correlation)
    - SIFT descriptor
    - location
- Edges, Lines
  - Similarity measured in terms of:
    - orientation
    - contrast
    - coordinates of edge or line's midpoint
    - length of line

Example: Comparing lines

l_l and l_r: line lengths

θ_l and θ_r: line orientations

(x_l,y_l) and (x_r,y_r): midpoints

c_l and c_r: average contrast along lines

ω_l ω_θ ω_m ω_c : weights controlling influence

$$S = \frac{1}{\omega_l(l_l - l_r)^2 + \omega_\theta(\theta_l - \theta_r)^2 + \omega_m[(x_l - x_r)^2 + (y_l - y_r)^2] + \omega_c(c_l - c_r)^2}$$

The more similar the lines, the larger S is!

FEATURE_MATCHING Algorithm

Inputs:

- I_l and I_r
- Set of features on the left and right

Things that must be chosen:

- Search Window
- Similarity measure

FEATURE_MATCHING Algorithm

For each feature f_l in the left image:

- Compute the similarity measure between f_l and every feature in the search window $R(f_l)$
- Select the feature in $R(f_l)$ that maximizes the similarity measure.
- Save the correspondence and the disparity of f_l

Output the list of correspondences and disparities.

Which method should we use?

Correlation methods:

- dense maps, good for surface reconstruction
- Require textured images
- Sensitive to illumination variations
- Inadequate for very different viewpoints

Feature methods:

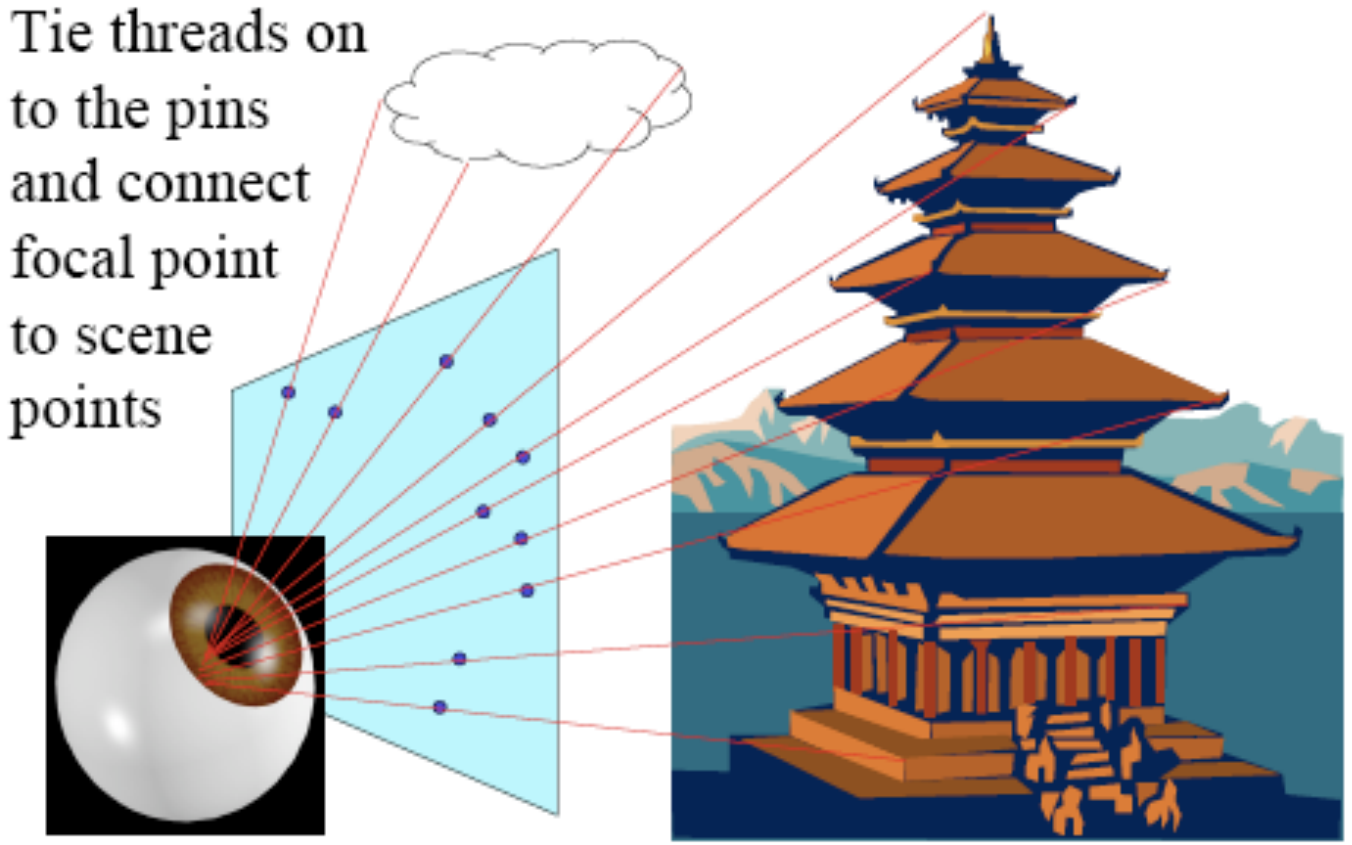
- Sparse maps, good for navigation
- Require prior knowledge of type of scene
- Must find features first

Constraining the Search Space

Finding correspondences is a search problem.

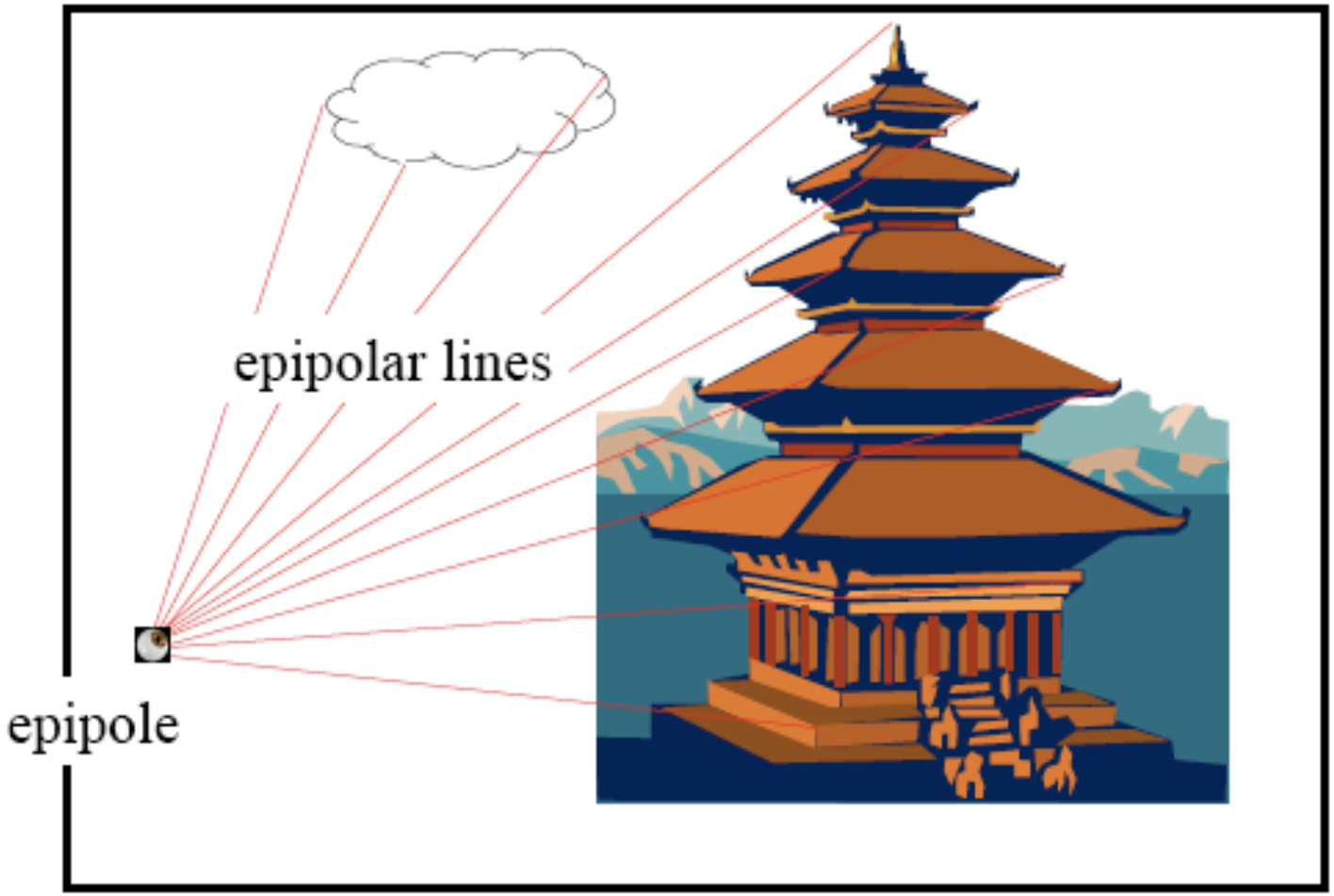
Geometry can be used to constrain the search.

Rays to Pts in the Scene



Now what would this look like to a second observer?

Rays Seen from Second Observer



Rays Seen by the First Viewer

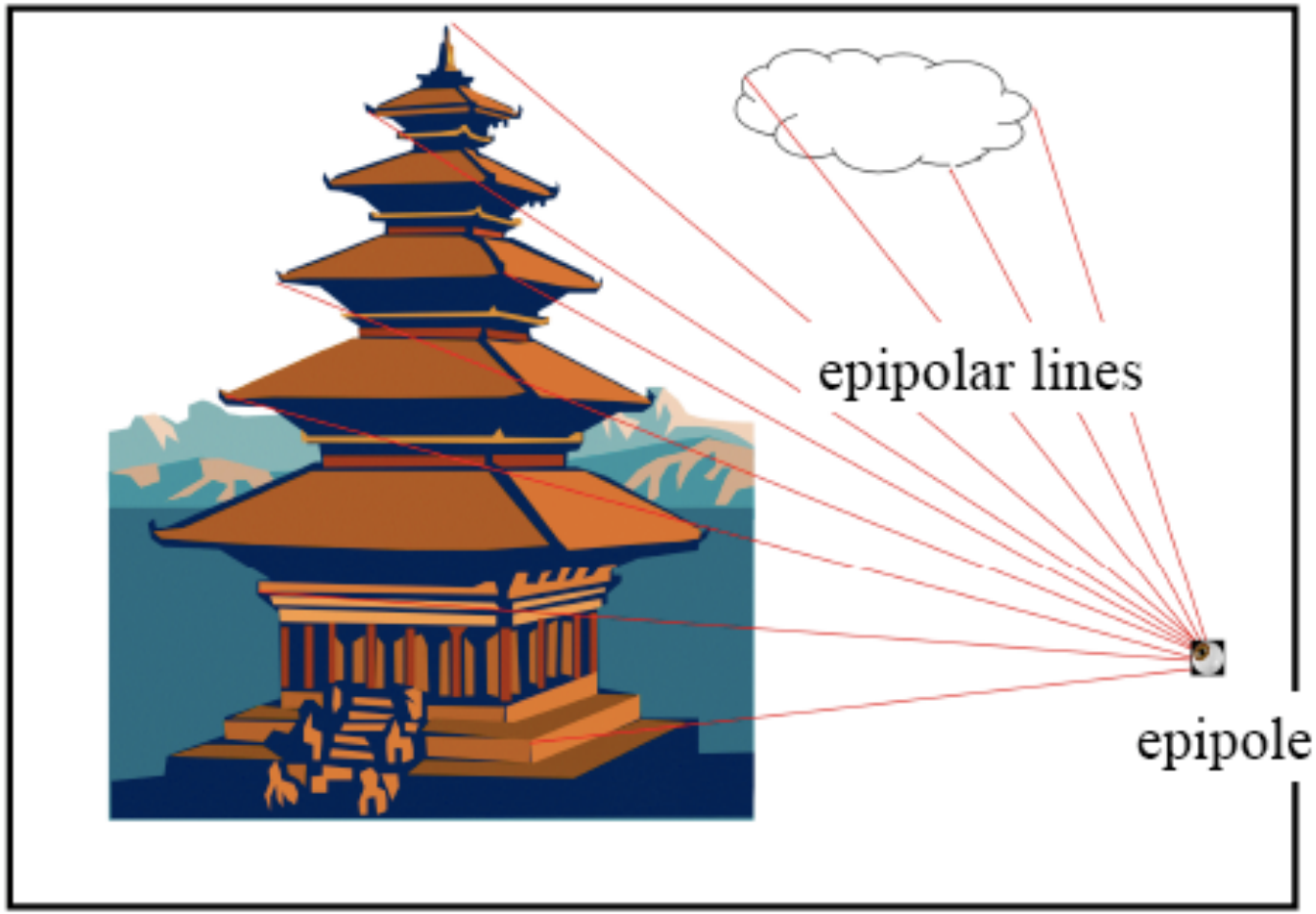
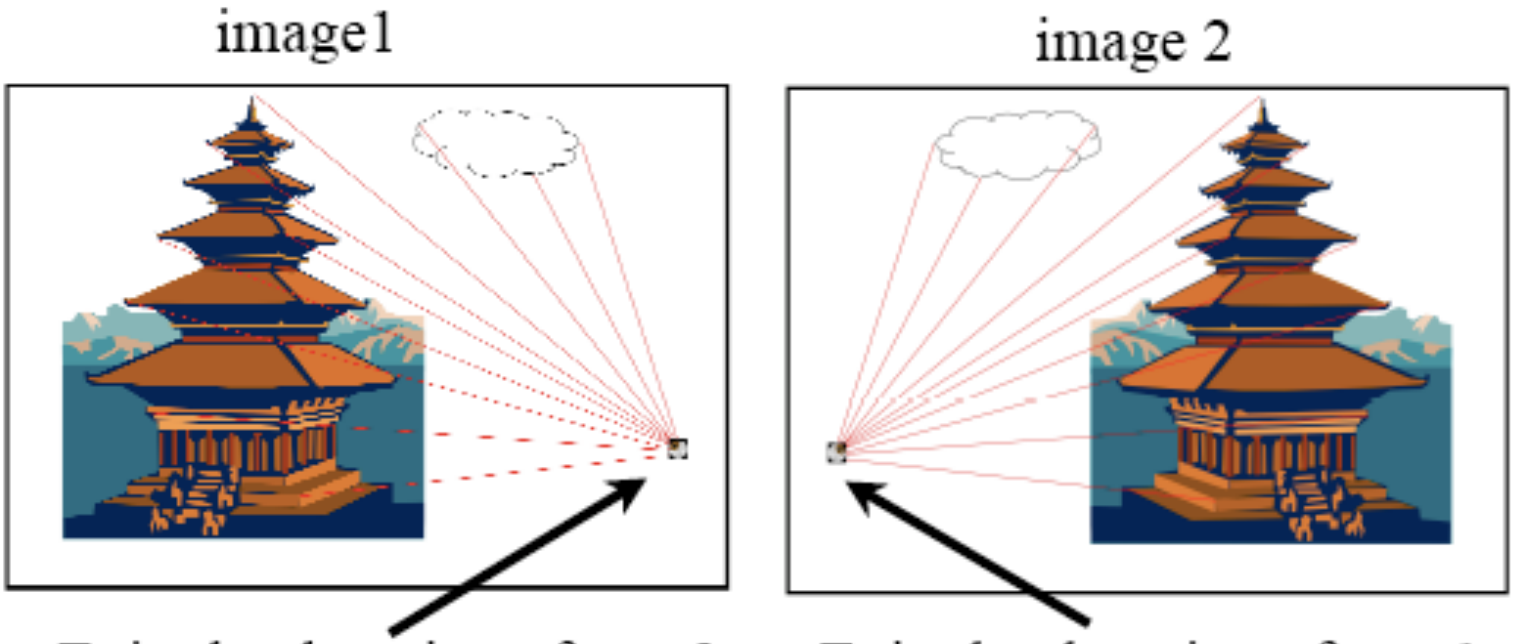
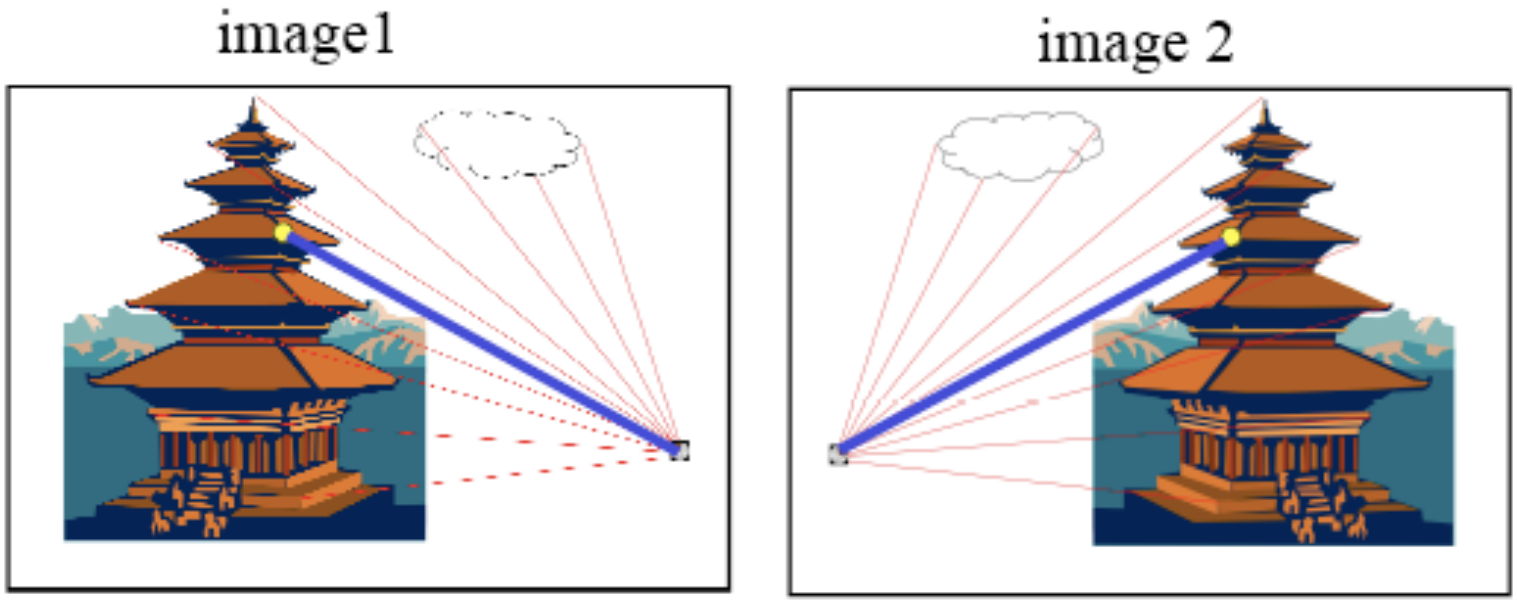


Image 1

Epipolar Geometry

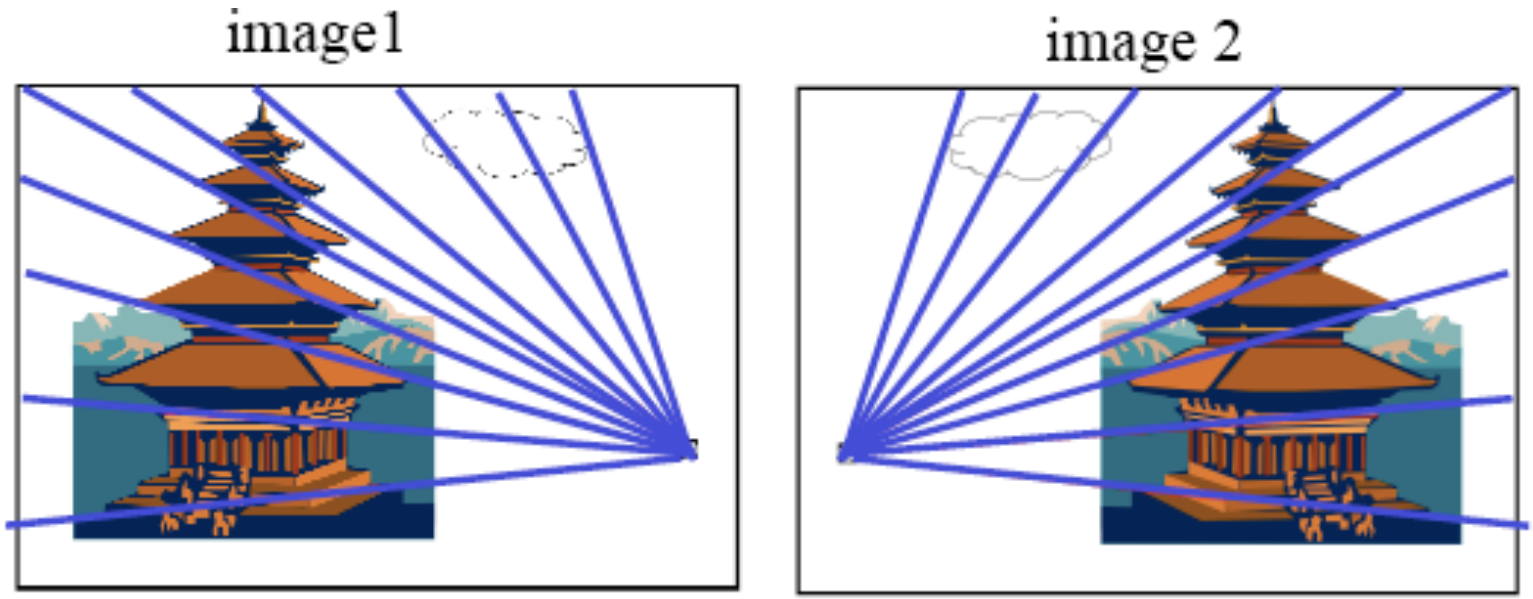


Epipolar Geometry



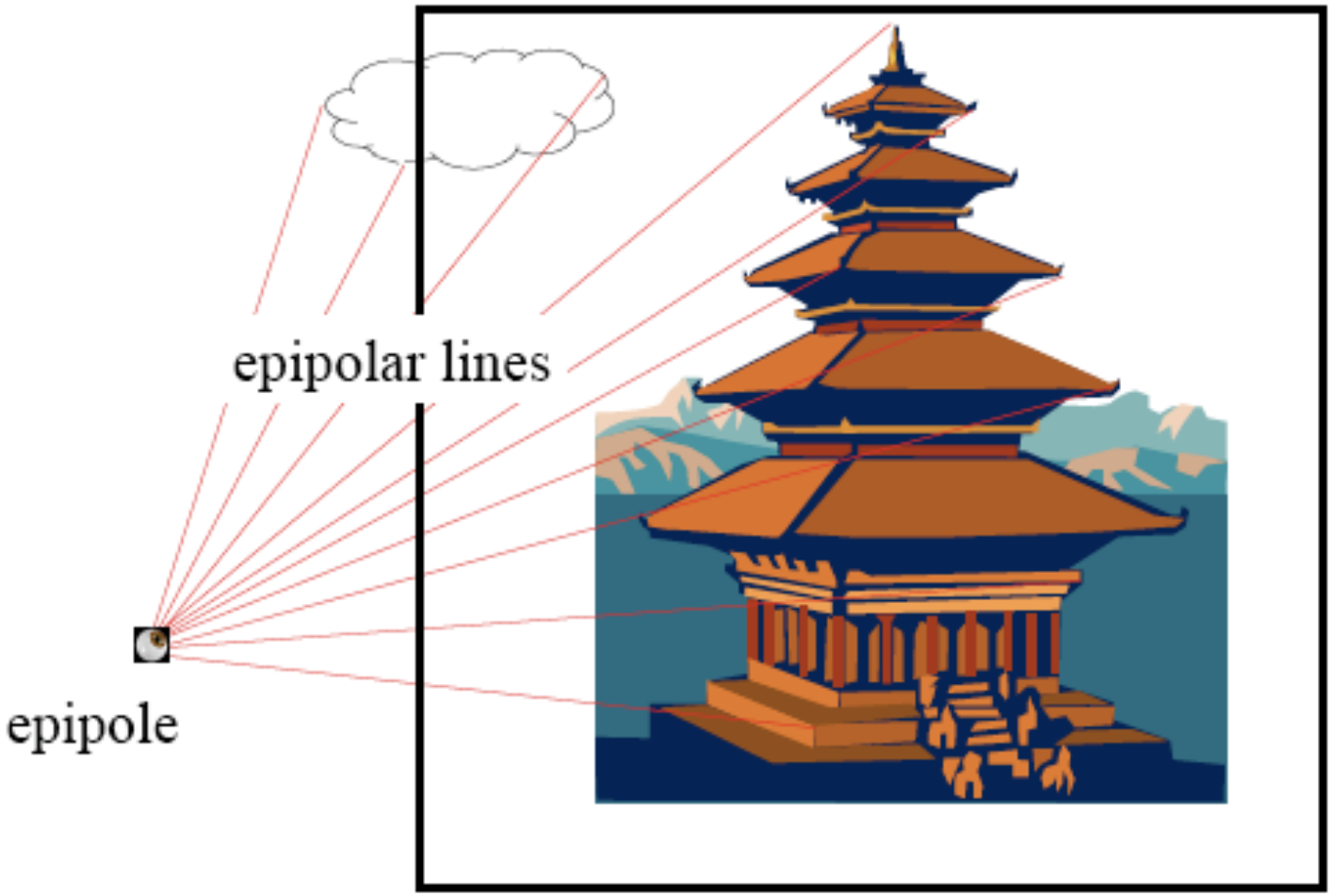
Corresponding points
lie on conjugate epipolar lines

Epipolar Geometry

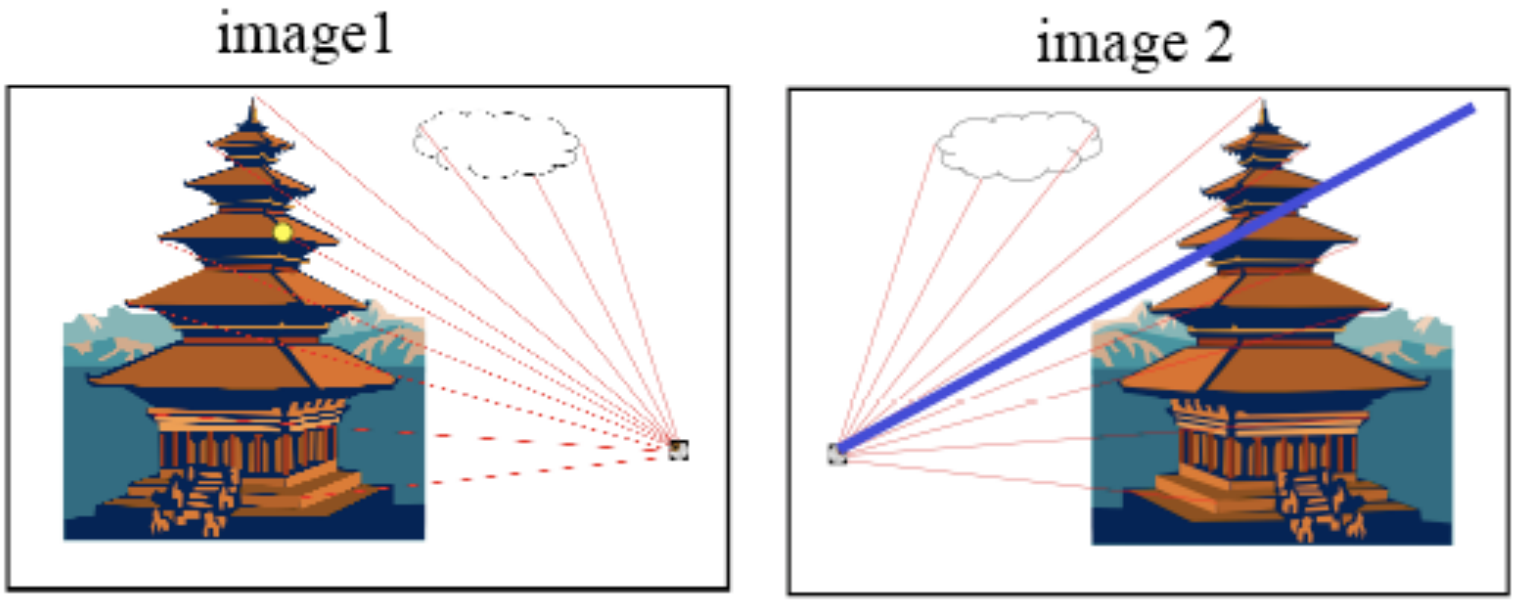


Conjugate epipolar lines induce
a generalized 1D "scan-line" ordering
on the images (analogous to traditional
scan line ordering of rows in an image)

Epipole, not necessarily in the image



How do we find Epipolar Lines?



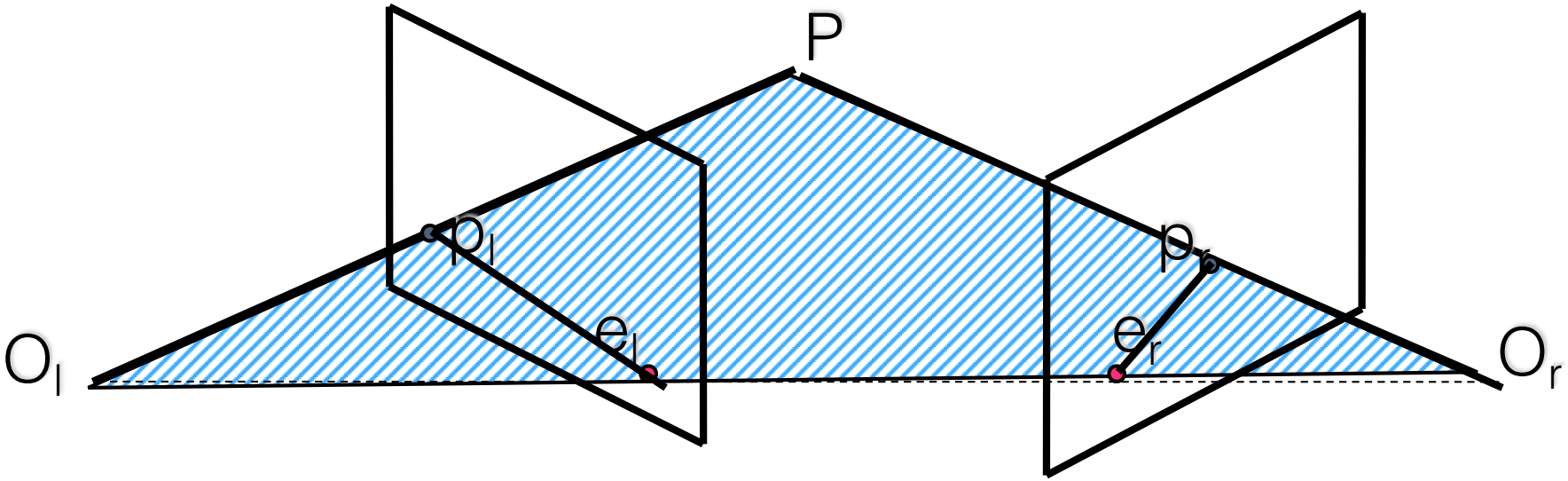
Given a point in one image, how do we determine the corresponding epipolar line to search along in the second image?

Essential Matrix

The ESSENTIAL matrix is a 3x3 matrix that "encodes" the epipolar geometry of two views.

Given a point in an image, multiplying by the Essential Matrix, will tell us the EPIPOLAR line in the second image where the corresponding point must be.

Epipolar Geometry



Epipoles:

- e_l: left image of O_r
- e_r: right image of O_l

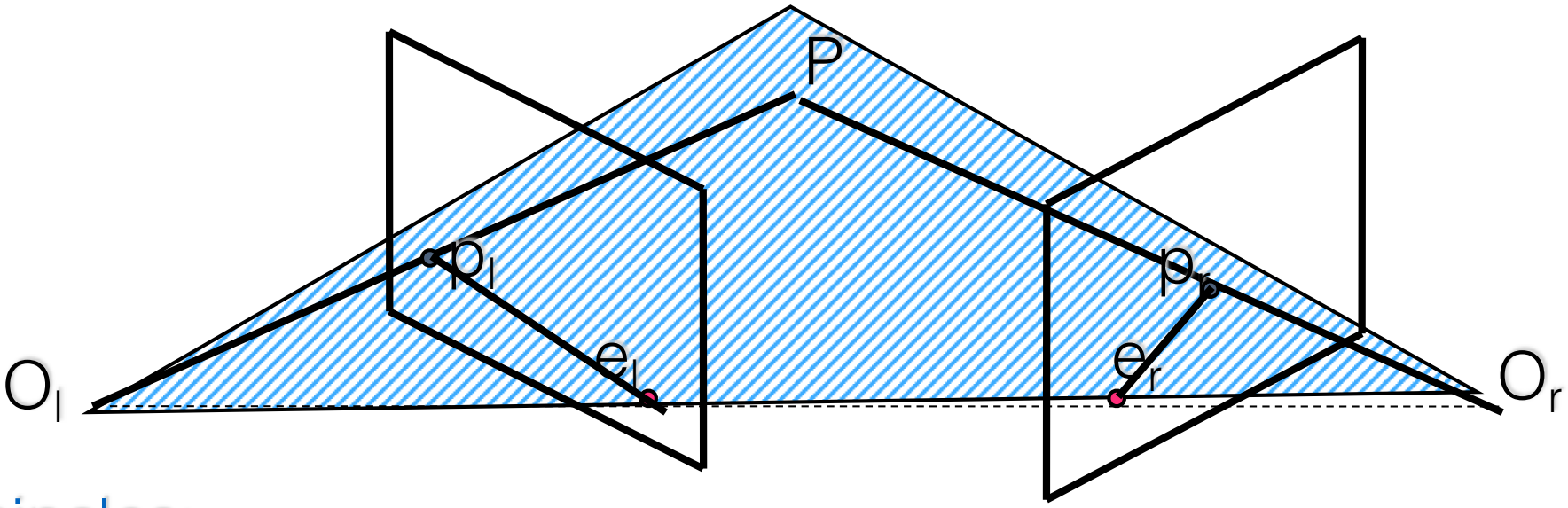
Epipolar plane:

- Three points: O_l,O_r, and P define an epipolar plane

Epipolar lines and epipolar constraint:

- Intersections of epipolar plane with the image planes
- Corresponding points are on "conjugate" epipolar lines

Epipolar Constraint:



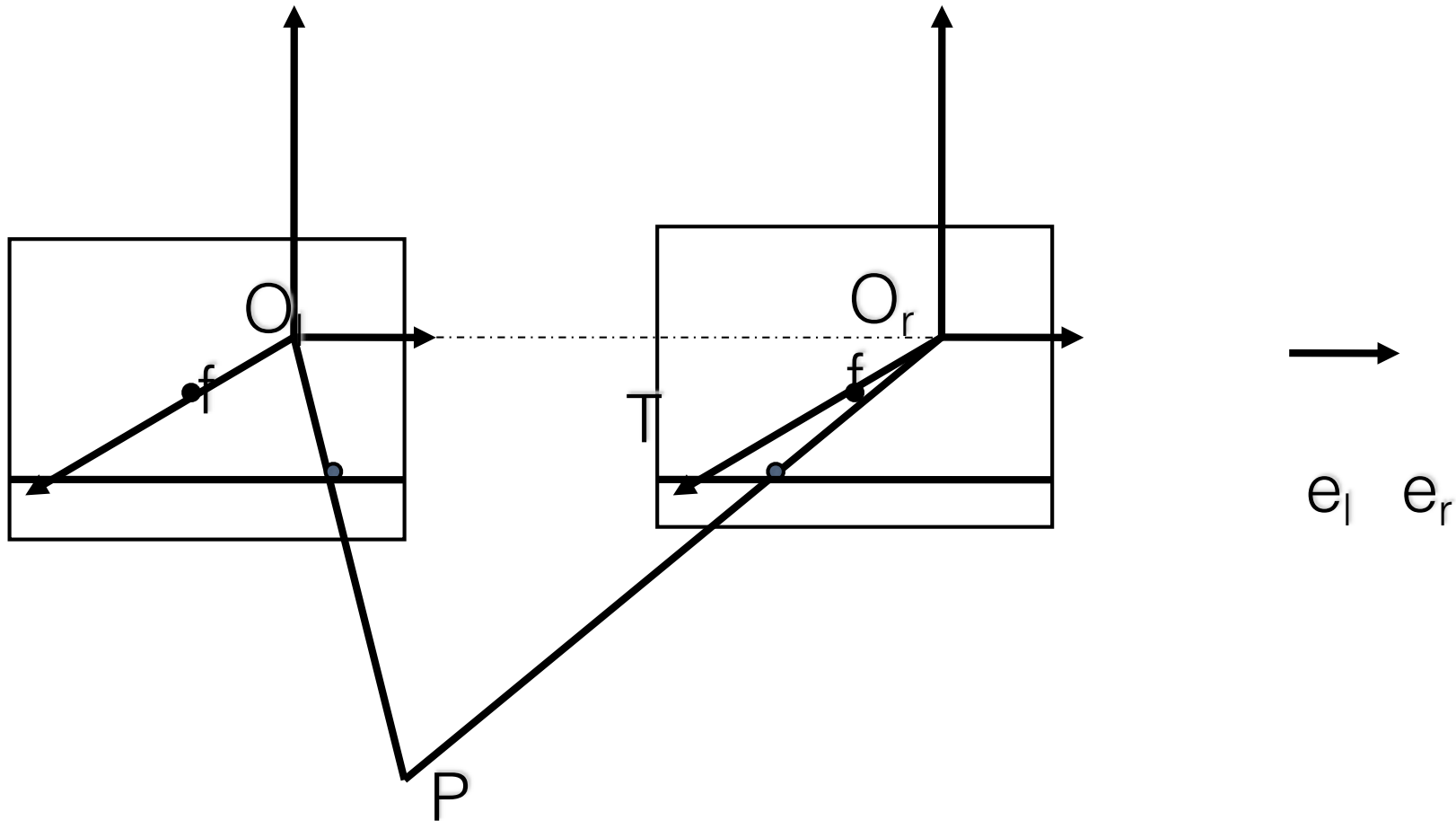
Find Epipoles:

- e_l: left image of O_r
- e_r: right image of O_l

Given p_l:

- consider its epipolar line: p_l e_l
- find epipolar plane: O_l,p_l,e_l
- intersect the epipolar plane with the right image plane
- search for p_r on the right epipolar line

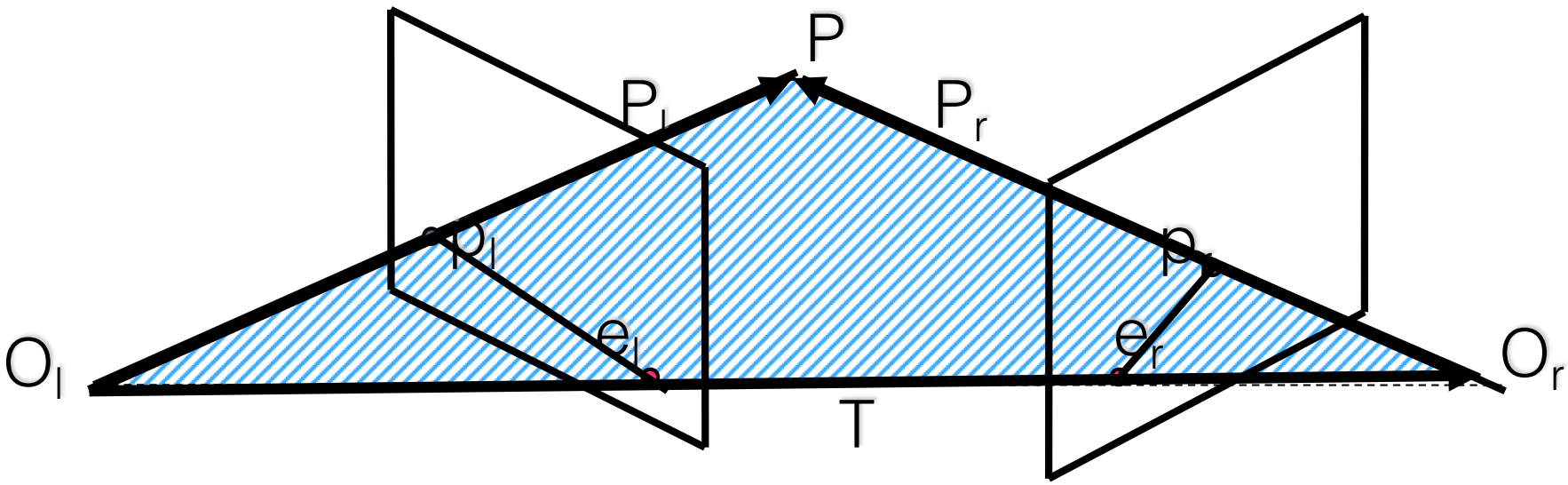
Epipolar Geometry for Parallel Cameras



Epipoles are at infinity

Epipolar lines are parallel to the baseline

Essential Matrix

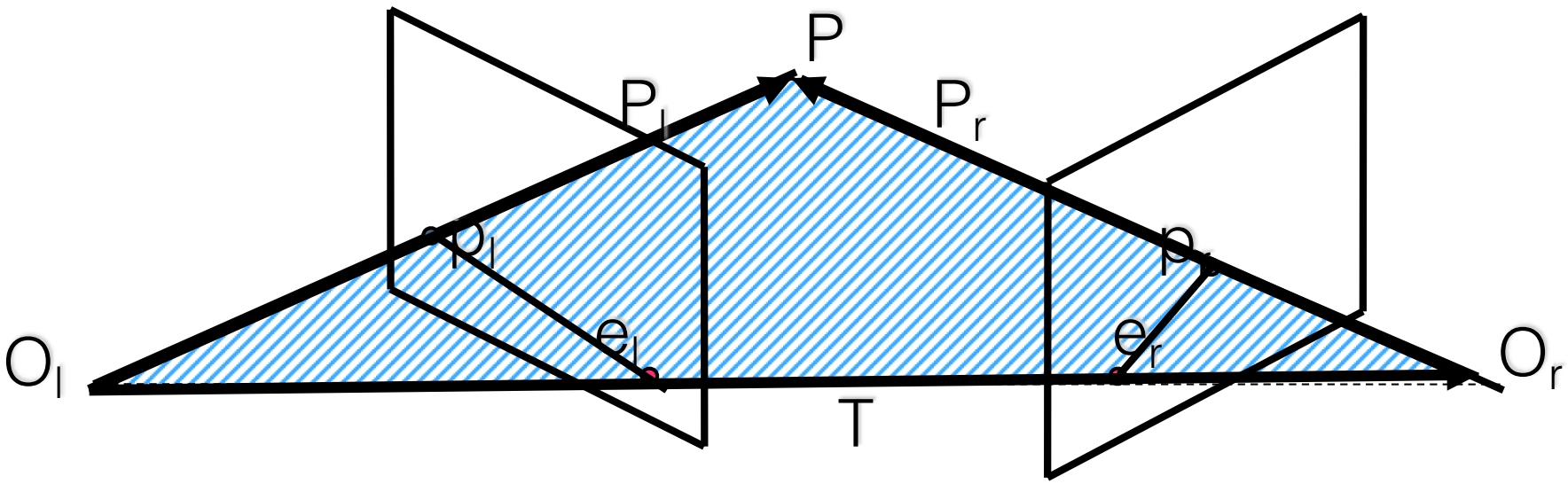


$$P_r = R(P_l - T)$$

$$P_l - T = R^{-1} P_r = R^T P_r$$

Essential Matrix

Epipolar constraint: P_l, T and P_l - T are coplanar:

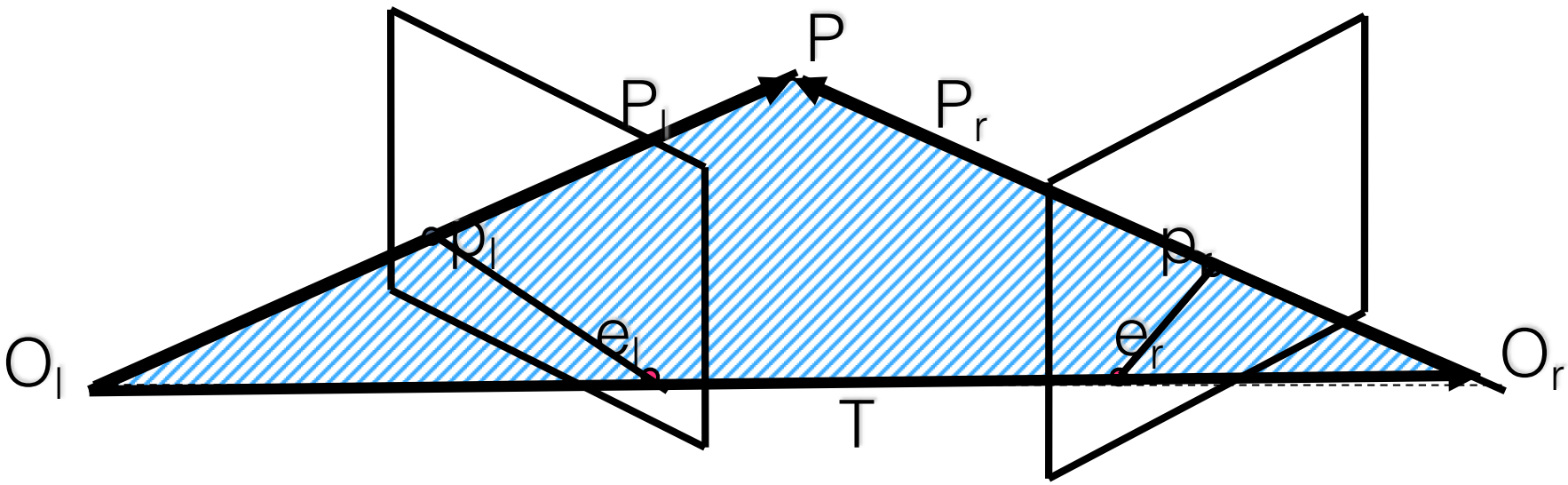


$$(P_l - T)^T \cdot T \times P_l = 0$$

$$P_l - T = R^T P_r \Rightarrow (R^T P_r)^T \cdot T \times P_l = 0$$

Essential Matrix

Epipolar constraint: P_l, T and P_l - T are coplanar:



$$(R^T P_r)^T \cdot T \times P_l = 0$$

$$(P_r^T R) \cdot (T \times P_l) = 0$$

Vector Product as a Matrix Multiplication

$$T \times P_l = \begin{vmatrix} i & j & k \\ T_x & T_y & T_z \\ P_{l_x} & P_{l_y} & P_{l_z} \end{vmatrix}$$

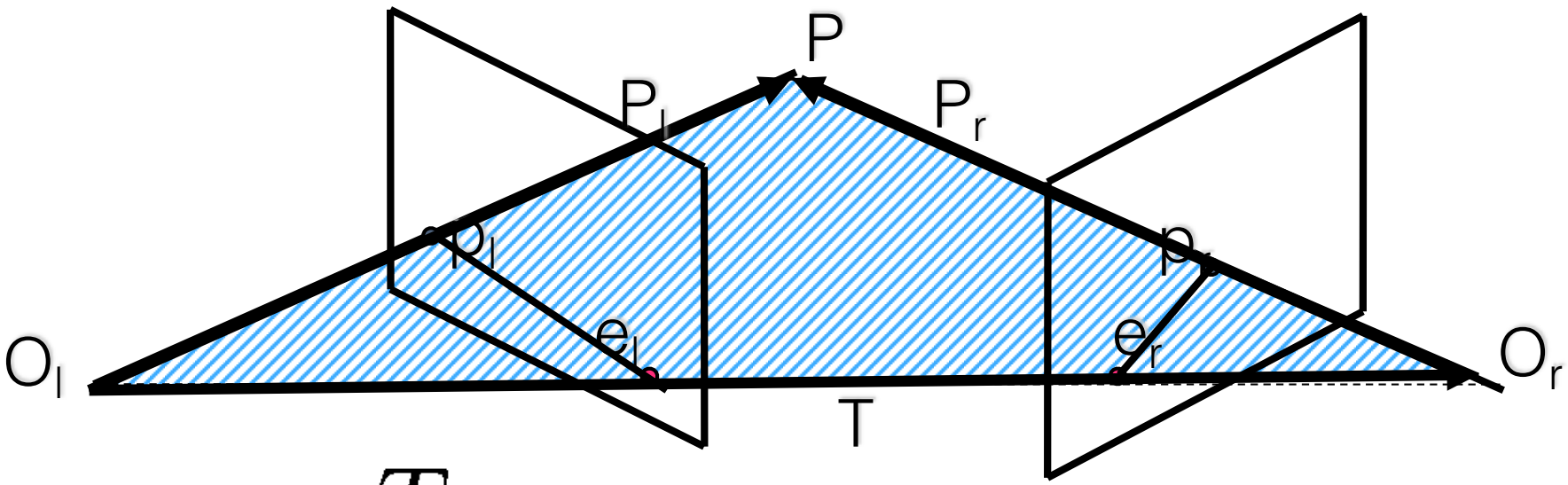
$$T \times P_l = (T_y P_{l_z} - T_z P_{l_y}) i + (T_z P_{l_x} - T_x P_{l_z}) j + (T_x P_{l_y} - T_y P_{l_x}) k$$

$$T \times P_l = S P_l = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \begin{bmatrix} P_{l_x} \\ P_{l_y} \\ P_{l_z} \end{bmatrix} = \begin{bmatrix} T_y P_{l_z} - T_z P_{l_y} \\ T_z P_{l_x} - T_x P_{l_z} \\ T_x P_{l_y} - T_y P_{l_x} \end{bmatrix}$$

S has rank 2 ; it depends only on T

Essential Matrix

Epipolar constraint: P_l, T and P_l - T are coplanar:

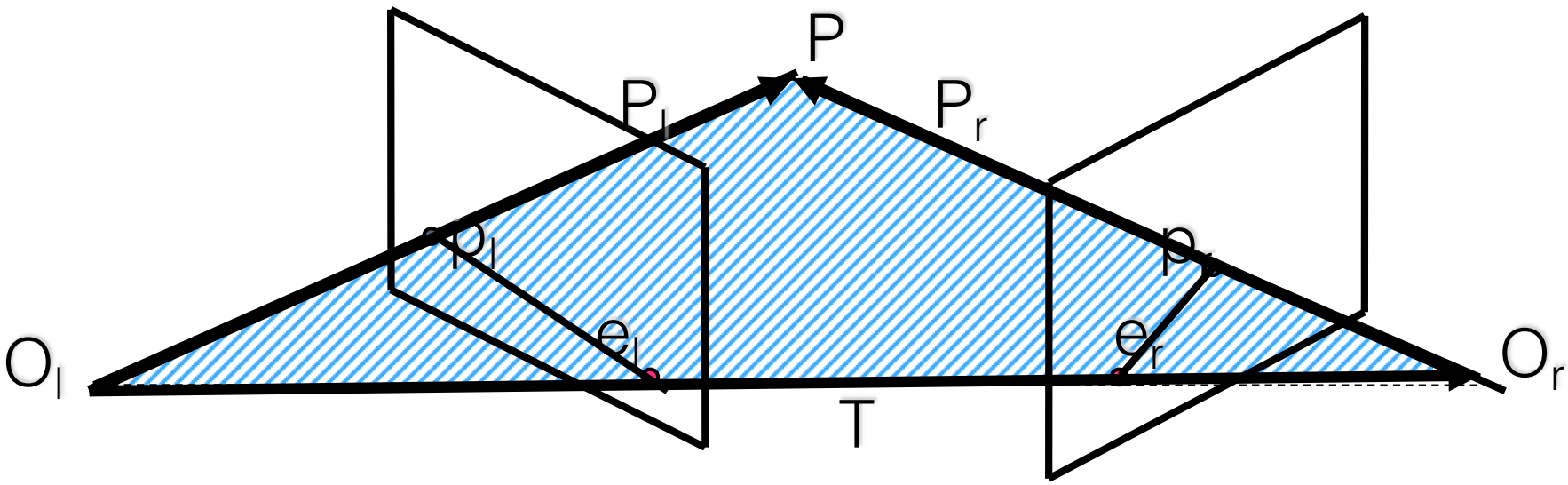


$$(P_r^T R) \cdot (T \times P_l) = 0$$

$$P_r^T R S P_l = 0$$

Essential Matrix

Epipolar constraint: P_l, T and P_l - T are coplanar:



$$P_r^T R S P_l = 0$$

Essential Matrix:

$$E = RS \qquad P_r^T E P_l = 0$$

Essential Matrix Properties

has rank 2

depends only on the EXTRINSIC Parameters (R & T)

$$E = RS$$